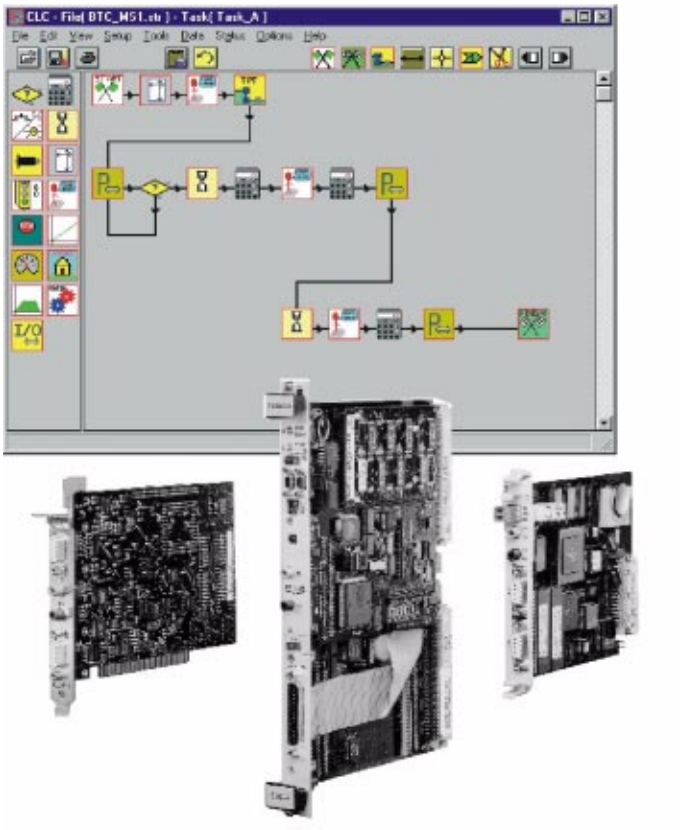
engineering
mannesmann
Rexroth



VisualMotion GPS 6.0

Reference Manual

DOK-VISMOT-VM*-06VRS**-FKB1-AE-P

Indramat

Title VisualMotion GPS 6.0

Kind of documentation Reference Manual

Docu-type DOK-VISMOT-VM*-06VRS**-FKB1-AE-P

Internal file reference IAE 74792, Rev. G

Purpose of this document This document is intended to be a reference manual for the CLC Multi-Axes Coordinated Motion Control Card. This manual contains the information needed to program the card using VisualMotion32 and covers the following items.

- I/O Systems
- Parameters
- Icon/Text Programming
- Direct ASCII Communication
- DDE Client Interfaces

Revision history

Revision	Date	Remarks
IAE 74792, Rev A	10/91	Initial Release
IAE 74792, Rev B	06/94	Updates
IAE 74792, Rev C	10/94	Updates - Not released
IAE 74792, Rev D	12/94	GPS 1.2
IAE 74792, Rev E	9/95	GPS 02
IAE 74792, Rev F	10/96	GPS 05
IAE 74792, Rev G	06/98	VisualMotion32 - GPS 06

Copyright © INDRAMAT, 1998

Copying this document, and giving it to others and the use or communication of the contents thereof without express authority, are forbidden. Offenders are liable for the payment of damages. All rights are reserved in the event of the grant of a patent or the registration of a utility model or design (DIN 34-1).

Validity All rights are reserved with respect to the content of this documentation and the availability of the product.

Published by INDRAMAT • 5150 Prairie Stone Parkway • Hoffman Estates, IL 60616

Telephone 847-645-3600 • Fax 847-645-6201

TechDoc Dept. (LS/HK)

Contents

1 Introduction 1-1

1.1 Purpose of Manual 1-1
1.2 Manual Overview 1-6
1.3 Overview of CLC 1-7
CLC System Architecture 1-7
Indramat's VisualMotion® Programming Interface 1-8
CLC BTC06 1-8
CLC Operating System 1-9
1.4 CLC Motion Capabilities 1-10
Non-Coordinated Motion 1-10
Coordinated Motion 1-10
Electronic Line Shaft (ELS) 1-11

2 CLC Input/Output Systems 2-1

2.1 I/O Overview 2-1
2.2 Remote I/O Systems 2-2
2.3 I/O Mapper 2-3
I/O Mapper Ladder Logic Format 2-4
I/O Mapper Boolean Format 2-4
I/O Mapper Operators 2-5
I/O Mapper Considerations 2-5
2.4 I/O Bit Forcing 2-6
2.5 Reading and Writing Physical I/O 2-6
2.6 CLC Registers 2-7
Register 1: System Control 2-8
Registers 2-5: Task Control 2-10
CLC Cycle Control Considerations 2-13
Register 6: System Diagnostic Code 2-14
Registers 7-10: Task Jog Control 2-14
Registers 11-18, 209-240: Axis Control 2-17
Register 21: System Status 2-18
Registers 22-25: Task Status 2-19
Registers 27 and 28: Eagle Module Inputs/Outputs (CLC-V Only) 2-20
Register 29: ELS Master Control Register 2-21

Register 30: ELS Master Status Register 2-22
Registers 31-38, 309-340: Axis Status 2-23
Registers 40-87: DEA (4/5/6) I/O 2-26
Registers 88,89: Task A Extend Event Control 2-26
Registers 90 and 91: Latch and Unlatch 2-27
Registers 92-94: Mask Pendant Key Functionality 2-27
Registers 95-97: BTC06 Teach Pendant Status Registers 2-28
Registers 98, 99: Teach Pendant Control - Task A-B, C-D 2-29
Register 400-405, 410-415: DEA (28/29/30) I/O Registers 2-30
CLC Reserved Register Tables 2-31

3 Parameters 3-1

3.1 Overview 3-1
3.2 Parameter Transfer Commands 3-2
3.3 List of Parameters 3-3
3.4 System Parameters 3-14
System Setup (0001-0031) 3-14
Jogging and Display (0042-0056) 3-24
Program Management (0090-0099) 3-26
System Status (0100-0126) 3-29
Electronic Line Shaft Parameters 1 of 2 (0150-0170) 3-33
VME Bus (0200-0287) 3-37
BTC06 Teach Pendant (0801-0991) 3-44
Electronic Line Shaft Parameters 2 of 2 (1000-1568) 3-53
System Parameter Lists (2000-3108) 3-59
3.5 Task Parameters 3-72
Task Setup (0001-0002) 3-72
Coordinated Motion (0005-0026) 3-74
Robotics (0035-0059) 3-78
Coordinated Motion Status (0100-0113) 3-79
Task Status (0120-0200) 3-81
Task Parameter Lists (2000-2001) 3-85
3.6 Axis Parameters 3-86
Axis Setup (0001-0038) 3-86
Axis Status (0100-0145) 3-101
Electronic Line Shaft (0150-0164) 3-104
Axis Feedback Capture (Registration 0170-0174) 3-108
Optional SERCOS Data (0180-0196) 3-109
Axis Parameter Lists (2000-2001) 3-111
3.7 Drive Parameters 3-111
Drive Status 3-113
Required DDS Setup 3-113

4 VisualMotion Menu Commands 4-1

4.1 Introduction ... 4-1
4.2 The File Menu ... 4-2
Program Management ... 4-2
Archive ... 4-4
Transfer Cams ... 4-5
Transfer Events ... 4-6
Transfer I/O Mapper ... 4-6
Transfer Parameters ... 4-6
Transfer Points ... 4-7
Transfer Variables ... 4-8
Transfer Zones ... 4-8
Print ... 4-8
4.3 The Edit Menu ... 4-10
Clear Current Task ... 4-10
Find, Find Next ... 4-10
Add Subroutine ... 4-11
Add Event Function ... 4-11
Labels - User Labels ... 4-11
Labels - Register Labels ... 4-13
Labels - Bit Labels ... 4-14
Import User Label File ... 4-15
Export User Label File ... 4-15
4.4 The View Menu ... 4-16
Subroutines ... 4-16
Event Functions ... 4-16
Zoom Out ... 4-16
4.5 The Setup Menu ... 4-17
Card Selection ... 4-17
Configuration ... 4-17
Drives ... 4-18
Drives Help Directories ... 4-18
Coordinated Motion ... 4-19
I/O Setup ... 4-20
Overview ... 4-22
Pendant Security ... 4-23
CLC Serial Ports ... 4-24
VME Configure ... 4-24
4.6 The Tools Menu ... 4-26
Breakpoint Control ... 4-26
CAM Builder ... 4-26
Jogging ... 4-31

Oscilloscope 4-34
VisualMotion32, CLC_DDE Release6 4-34
Show Program Flow 4-35
4.7 The Data Menu 4-36
CAM Indexer 4-36
Events 4-36
Field Bus Mapper (CLC-D Only) 4-38
I/O Mapper 4-41
PID Control Loops 4-45
PLS 4-51
Points 4-58
Registers 4-60
Registration (Not Functional for GPS 6.0) 4-63
Sequencer 4-64
Variables 4-67
Zones 4-68
4.8 The Status Menu 4-69
Diagnostic Log 4-69
Drives 4-69
Drives on Ring 4-69
System 4-70
Tasks 4-70
4.9 The Options Menu 4-72
4.10 Help 4-73

5 Programming Concepts 5-1

5.1 Overview 5-1
5.2 Program Tasks 5-1
Command Execution 5-1
5.3 Events 5-2
Time-based Events 5-2
Distance-based Events 5-3
Repeating Axis Position (Rotary) Events 5-5
Interrupt Input Events 5-6
Extended External I/O Events 5-7
VME Events 5-7
Feedback Capture (Probe) 5-9
Event Tables 5-10
5.4 Subroutines 5-10
Function Arguments and Local Variables 5-10
5.5 Sequencer 5-11
Single Stepping a Sequencer Step 5-12
Single Stepping a Step Function 5-12

5.6 Data and Expressions ... 5-13
Integers and Floats ... 5-13
Constants and Variables ... 5-13
Expressions ... 5-15
Mathematical and Logical Operators ... 5-16
5.7 Tables ... 5-17
Event Table ... 5-19
CLC Path Planner ... 5-20
Absolute Point Table ... 5-22
Relative Point Table ... 5-23
Zone Protection Table ... 5-24

6 Icon Programming 6-1

6.1 Introduction ... 6-1
Working with VisualMotion's Icon Palettes and Buttons ... 6-1
Connecting Icons ... 6-3
Icon Labels and Comments ... 6-5
User Defined Labels ... 6-5
6.2 CLC Icons ... 6-6
Single Axis Icon Palette ... 6-6
Coordinated Motion Icon Palette ... 6-7
ELS Icon Palette ... 6-8
Utility Icon Palette ... 6-9
Accel ... 6-10
Axis ... 6-11
AxisEvt ... 6-16
Branch ... 6-17
Calc ... 6-19
Cam ... 6-22
CamAdj ... 6-23
CamBuild ... 6-24
Cam Indexer ... 6-26
Circle ... 6-31
Decel ... 6-33
ELS ... 6-34
ELSAdj ... 6-37
ELSMode ... 6-38
Event ... 6-39
Finish ... 6-40
Go ... 6-41
Home ... 6-42
I/O ... 6-43
Join ... 6-44

Joint ... 6-45
Line ... 6-46
Move ... 6-47
Msg ... 6-48
Param ... 6-49
Path ... 6-51
PathCalc ... 6-52
Path/Hold ... 6-53
PID ... 6-54
PLS ... 6-56
Position ... 6-57
PrmBit ... 6-58
Prmint ... 6-59
Probe ... 6-60
ProbeEvt ... 6-61
Ratio ... 6-62
Register Transfer ... 6-64
Registration (Not Functional with GPS 6.0) ... 6-65
RobotOrg ... 6-72
RobotTool ... 6-72
Scissor ... 6-73
Sequencer ... 6-74
Size ... 6-76
Start ... 6-77
Stop ... 6-78
Sub ... 6-79
Velocity ... 6-80
VME ... 6-81
VMEEvt ... 6-83
Wait ... 6-84

7 Text Language Programming 7-1

7.1 Overview ... 7-1
7.2 Directives ... 7-1
EQU (Equate) ... 7-1
DEFINE ... 7-2
EVENT/START & EVENT/END ... 7-2
TASK/START & TASK/END ... 7-3
7.3 CLC Instructions ... 7-3
Instruction Format ... 7-3
AXIS/EVENT ... 7-6
AXIS/HOME ... 7-7
AXIS/INITIALIZE ... 7-8

AXIS/MOVE (Single Axis, Non-Coordinated) ... 7-10
AXIS/RATIO ... 7-11
AXIS/SPINDLE (Continuous Velocity Mode) ... 7-13
AXIS/START ... 7-15
AXIS/STOP ... 7-16
AXIS/WAIT (Axis Wait For In-Position) ... 7-17
CALL ... 7-18
CAM/ACTIVATE ... 7-19
CAM/ADJUST ... 7-20
CAM/BUILD ... 7-21
CAM_INDEX Command ... 7-22
CAM/STATUS Command ... 7-23
CAPTURE/ENABLE ... 7-24
CAPTURE/SETUP ... 7-25
DATA/SIZE (Configure Memory Allocations) ... 7-26
DEFINE (Define Name For Variable) ... 7-28
DELAY (Suspend Task Execution) ... 7-29
ELS/ADJUST (Adjust ELS Axis) ... 7-30
ELS/DECEL (Set Master Decel) ... 7-31
ELS/INIT (Initialize ELS Axes) ... 7-32
ELS/MODE (Set ELS Axis Mode) ... 7-34
ELS/STOP ... 7-35
EVENT/DONE (Signal Event Completed) ... 7-36
EVENT/END (Mark End of Event) ... 7-38
EVENT/START (Start of Event function) ... 7-39
EVENT/TRIGGER (Trigger a Task Event) ... 7-40
EVENT/WAIT (Pause Task for Event Done Signal) ... 7-42
FUNCTION/ARG ... 7-43
FUNCTION/END ... 7-44
FUNCTION/START ... 7-45
GOSUB (Go To Subroutine) ... 7-46
GOTO (Go To Mark) ... 7-47
IF (If-Else-Endif Conditional Branch) ... 7-48
KINEMATIC (Use a Kinematic Definition for a Task) ... 7-53
LOCAL/VARIABLE ... 7-54
MESSAGE/DIAG (Task Diagnostic Message Definition) ... 7-55
MESSAGE/STATUS (Task Status Message Definition) ... 7-56
MOVE/CIRCLE (Coordinated Move with Circular Interpolation) ... 7-57
MOVE/JOINT (Coordinated Move Joint Point to Point) ... 7-58
MOVE/LINE (Coordinated Move with Straight Line Interpolation) ... 7-59
PARAMETER/BIT (Initialize Parameter Bit) ... 7-60
PARAMETER/GET (Load Parameter to a Variable) ... 7-62
PARAMETER/INIT (Initialize a Parameter) ... 7-63

PARAMETER/SET (Set a Parameter) ... 7-64
PATH/ABORT (Aborts Coordinated Motion) ... 7-65
PATH/POSITION (Get Current Path Absolute Position) ... 7-66
PATH/RESUME (Resume Coordinated Motion) ... 7-67
PATH/STOP (Halt Coordinated Motion) ... 7-68
PATH/WAIT (Pause Program for Motion) ... 7-69
PID/CONFIGURE ... 7-70
PLC/CLEAR (Clear I/O Register Bit) ... 7-71
PLC/READ (Read I/O Register(s)) ... 7-72
PLC/SET (Set I/O Register Bit) ... 7-73
PLC/TEST (Test I/O Register Bit) ... 7-74
PLC/WAIT (Pause Program for I/O) ... 7-75
PLC/WRITE (Write to I/O Register(s) ... 7-76
PLS/INIT (Compiler directive) ... 7-77
REGISTRATION ... 7-79
RETURN (Return From Subroutine) ... 7-80
ROBOT/ORIGIN ... 7-80
ROBOT/TOOL ... 7-81
SEQUENCER ... 7-82
SEQ/LIST ... 7-83
SEQ/STEP ... 7-84
TASK/AXES (Task Axes Definition) ... 7-85
TASK/END (Mark the End of a Task) ... 7-87
TASK/START (Define the Start of a Task(s)) ... 7-88
VAR/INIT ... 7-89
VME/BROADCAST (Broadcast Event to VME) ... 7-90
VME/EVENT (Send Event to VME Unit) ... 7-91
VME/READ (VME Bus Data Read) ... 7-92
VME/WRITE (VME Bus Data Write) ... 7-94

8 Program Debugging and Monitoring 8-1

8.1 Finding Program Problems ... 8-1
Test Code ... 8-2
8.2 CLC Compiler Base Code ... 8-2
Base Code instruction mnemonics and valid arguments ... 8-2
8.3 Icon Language Warnings and Error Messages ... 8-18
8.4 Text Language Error Messages ... 8-20
First Pass Errors ... 8-21
Second Pass Compiler Errors ... 8-21

A Direct ASCII Communication A-1

A.1 Overview ... A-1

A.2 CLC Communication Protocol A-1
Reading Data from the CLC / TRANS 01-D A-2
Writing Data to the CLC / TRANS 01-D A-2
Communication Errors A-2
Checksum A-3
End of Message A-3
Backspaces and White Spaces A-4
Numeric Data Formats A-4
Format of Data Sent to the CLC / TRANS 01-D A-4
A.3 Command Classes/Subclasses A-5
Parameters A-5
Variables A-5
PID A-6
Point Tables A-6
Event Tables A-7
Program Communication A-7
Functions A-7
I/O Registers A-7
Sequencer Data A-8
PLS A-8
Zones A-8
A.4 Drive and CLC / TRANS 01-D Parameters and Subclasses A-9
Parameter Data Subclass A-9
Name Text Subclass A-9
Units Text Subclass A-9
Upper Limit, L: Lower Limit Subclasses A-9
Attribute Subclass A-10
Parameter Lists Subclasses A-10
SERCOS Parameter Sets A-10
A.5 Parameter Lists A-11
Listing a Parameter A-11
Parameter List Block Transfer A-12
A.6 User Program Variables A-15
'P': Data A-15
'T': Label Text A-15
A.7 PID A-16
Getting the assigned variables A-16
A.8 Point Tables A-17
Point Table Data A-18
Point Table Data, Row Format A-18
List of All User-defined Labels (V) A-19
A.9 Event Tables A-20
Event Table Data A-20

Event Table Data, Row Format A-21
A.10 CLC Program Communication A-22
Program Header Record A-22
Download Block Size A-22
Initializing a Download (PW) A-22
Executing a Download (PD) A-23
Initializing an Upload (PR) A-24
Executing an Upload (PD) A-24
Erasing (Deleting) a Program (PE) A-24
List CLC Resident Programs (PH) A-25
Activating a Program (PA) A-25
Request Currently Active Program (PA) A-26
Request Name of Program (PN) A-26
List of Event Function Marks (PF) A-26
List Variable Labels (PV) A-27
Transfer Tables Between Programs (PX) A-28
CAM Indexer function block variables (PJ) A-29
Sequencer/Subroutine Related Subclasses A-30
Registration Block Information A-30
A.11 Function Class (S) A-31
A.12 Input/Output Registers A-32
I/O Register Access (RB), (RX), (RD) A-33
Set Current I/O State with Mask (RM) A-33
I/O Forcing Selection (RF) A-34
I/O Forcing State Change (RC) A-34
I/O Binary Forcing State (RS) A-35
Erase All Forcing Masks (RE) A-35
A.13 Sequencer Data A-36
Sequence List Class (L) A-36
Sequence Table Class (Q) A-37
A.14 PLS A-39
Format of Configuration Data A-40
Format of Switch Values A-40
Drive PLS ASCII Protocol A-41

B DDE Client Interfaces B-1

B.1 Creating and Customizing a DDE Client Interface in Microsoft Excel™ B-1
DDE Worksheet Functions B-1
DDE Functions using Visual Basic® for Excel™ B-3
B.2 Wonderware B-7
Tagnames B-9

C ELS Configuration C-1

C.1 Electronic Line Shafting (ELS) Overview C-1
C.2 ELS Masters C-1
Real Masters C-1
Virtual Masters C-2
Second Master C-2
C.3 ELS Slaves C-2
Phase Synchronization C-3
Velocity Synchronization C-4
CAM Synchronization C-5
Dynamic synchronization C-5
ELS Master Synchronization C-7
Slave Secondary Mode (Non-Dynamic Synchronization) C-9
Summary of ELS Parameters and Registers C-9
C.4 Cam Applications C-10
Overview C-10
Cam Commands C-10
C.5 Building Cam Tables C-10
PCAM Cam Build Type C-12
VCAM Cam Build Type C-13
ACAM Cam Build Type C-15
SCAM Cam Build Type C-16
C.6 Cam Equation C-17
Consider H=0 and L≠0, Sph=0 ..... C-17
Consider H≠0, L=0, M=N =1 and Mph=Sph=0 ..... C-18
Consider L=0, H=1, M=2, N=1 and Mph=Sp=0 ..... C-20
Consider L=0, H=M=N=1 and Mph=90, Sph=90 degrees. C-21
C.7 CLC Cam Alignment C-22
Aligned Status Bit (Axis Status Register bit 8) C-23
C.8 Drive Cams C-24
Drive Cam Equation C-24
Axis Configuration C-24
Drive Cam Table Parameters C-25
Drive Cam Alignment C-26
Phase Adjusted Status (Axis Status Register bit 4) C-27
Drive Cam Configuration Notes C-28
C.9 Differences between CLC and Drive Cams C-29
C.10 Application (Rotary Cutoff) C-30

D CLC Drive Parameter Editor D-1

D.1 Overview D-1
D.2 File Menu D-2
Transfer Parameters D-2

D.3 Parameter Menu D-3
Analog Outputs D-3
Drive Direction D-4
Drive Name D-4
Drive Monitoring D-5
Drive Tuning D-6
Drive Limits D-7
Drive Reference D-8
Mechanical D-10
Overview D-11
D.4 Oscilloscope D-13
File Menu D-13
Source Menu D-14
Timing D-14
Signal Selection D-14
Abort, Upload and Enable Trigger D-16

I Index **I-1**

1 Introduction

1.1 Purpose of Manual

This document is a reference for the users and programmers of the CLC motion control card. For information pertaining to other system components, refer to the following documentation:

DIAX03 ELS Documentation Supporting Version 50-04V-EN:

Title: DIAX03 Mappe Set (50-04V-EN) for Diax03 Drives with Electric Gear Functions

Type of Document : Documentation Set

Pub No. 209-0072-4301

Part No. 273438

Title: Diax03 Drive with Electric Gear Functions

Type of Document : Functional Description: ELS 04VRS

Pub No. 209-0072-4302

Part No. 273442

Title: Diax03 Drive with Electric Gear Functions

Type of Document : Drive Configuration: ELS 04VRS

Pub No. 209-0072-4303

Part No. 273440

Title: DIAX03 Help System for Version 50-04V-EN

Type of Document : 1.44 MB Help Disks

Pub No. Not Applicable

Part No. 273439

Title: Diax03 Drive with Electric Gear Functions

Type of Document : Trouble Shooting Guide: ELS 04VRS

Pub No. 209-0072-4304

Part No. 273441

DIAX03 ELS Documentation Supporting Version 50-05V-EN:

Title: DIAX03 Drive With Electric Gear Function

Type of Document : Mappe Set

Pub No. 209-0072-4321

Part No. 276238

Title: DIAX03 Drive With Electric Gear Function	**Type of Document :** Trouble Shooting Guide: ELS 05VRS
Pub No. 209-0072-4322	**Part No.** 276242
Title: DIAX03 Drive With Electric Gear Function	**Type of Document :** Drive Configuration: ELS 05VRS
Pub No. 209-0072-4323	**Part No.** 276240
Title: DIAX03 Drive With Electric Gear Function	**Type of Document :** Trouble Shooting Guide: ELS 05VRS
Pub No. 209-0072-4324	**Part No.** 276241
Title: DIAX03 Drive With Electric Gear Function	**Type of Document :** Version Notes: ELS 05VRS
Pub No. 209-0072-4325	**Part No.** 276239
Title: DIAX03 Help System for Version 50-05V-EN	**Type of Document :** 1.44 MB Help Disks
Pub No. Not Applicable	**Part No.** 276237

DIAX03 SSE Documentation Supporting Version 51-02V-EN:

Title: DIAX03 Drive With Servo Function	**Type of Document :** Drive Configuration: SSE 02VRS
Pub No. 209-0072-4333	**Part No.** 276246
Title: DIAX03 Drive With Servo Function	**Type of Document :** Documentation Set
Pub No. 209-0072-4331	**Part No.** 276244
Title: DIAX03 Drive With Servo Function	**Type of Document :**
Pub No. 209-0072-4332	**Part No.** 276248
Title: DIAX03 Drive with Servo Function	**Type of Document :** Trouble Shooting Guide: SSE 02VRS

Pub No. 209-0072-4334 **Part No.** 276247

Title: DIAX03 Drive with Servo Function **Type of Document :** Version Notes

Pub No. 209-0072-4335 **Part No.** 276245

Title: DIAX03 Help System for Version 51-02V-EN **Type of Document :** 1.44 MB Help Disks

Pub No. Not Applicable **Part No.** 276243

DIAX04 ELS Documentation Supporting Version 60-05V-EN:

Title: Diax04 Drive with Electric Gear Functions **Type of Document :** Version Notes

Pub No. 209-0077-4325 **Part No.**

Title: Diax04 Drive with Electric Gear Functions **Type of Document :** Trouble Shooting Guidelines: ELS 05VRS

Pub No. 209-0077-4324 **Part No.**

Title: Diax04 Drive with Electric Gear Functions **Type of Document :** Functional Description: ELS 05VRS

Pub No. 209-0077-4322 **Part No.**

Title: DIAX04 Help System for Version 60-05V-EN **Type of Document :** 1.44 MB Help Disks

Pub No. Not Applicable **Part No.**

Title: DIAX04 Mappe Set (60-05V-EN) for Drives with Electric Gear Functions **Type of Document :** Documentation Set

Pub No. 209-0077-4321 **Part No.**

Title: Diax04 Drive with Electric Gear Functions **Type of Document :** Information, Selection Overview: ELS 05VRS

Pub No. 209-0077-4323 **Part No.**

DIAX04 SSE Documentation Supporting Version 61-02V-EN:

Title: DIAX04 Drive with Servo Function

Type of Document : Documentation Set

Pub No. 209-0077-4331

Part No. 276264

Title: DIAX04 Drive with Servo Function

Type of Document : Functional Description

Pub No. 209-0077-4332

Part No. 276272

Title: DIAX04 Drive with Servo Function

Type of Document : Information, Selection Overview

Pub No. 209-0077-4333

Part No. 276268

Title: DIAX04 Drive with Servo Function

Type of Document : Troubleshooting Guidelines

Pub No. 209-0077-4334

Part No. 276270

Title: DIAX04 Drive With Servo Function

Type of Document : Version Notes: SSE 02VRS

Pub No. 209-0077-4335

Part No. 276266

Title: DIAX04 Help System for Version 61-02V-EN

Type of Document : 1.44 MB Help Disks

Pub No. Not Available

Part No. 276263

ECODRIVE DKC2.1 SSE Documentation Supporting Version 57-03V-EN:

Title: ECODRIVE DKC 2.1 Drive Controller

Type of Document : Project Planning

Pub No. 209-0069-4388

Part No. 267473

Title: ECODRIVE Drive Controller

Type of Document : Functional

DKC 2.1	Description: SSE 03VRS
Pub No. 209-0073-4322	**Part No.** 274827
Title: ECODRIVE Drive Controller DKC 2.1	**Type of Document :** Trouble Shooting Guide: SSE 03VRS
Pub No. 209-0073-4324	**Part No.** 274828
Title: ECODRIVE Drive Controller DKC 2.1	**Type of Document :** Version Notes: SSE 03VRS
Pub No. 209-0073-4325	**Part No.** 274826
Title: Mappe 57-03V- EN: ECODRIVE DKC02.1 Drive Controller	**Type of Document :** Documentation Set
Pub No. 209-0073-4321	**Part No.** 274825
Title: Ecodrive DKC02.1 Help System for Version 57-03V-EN	**Type of Document :** 1.44 MB Help Disks
Pub No. Not Available	**Part No.** 274822

NOTE: *This manual is to be used in conjunction with the CLC Start Up Guide and End User Manual. Contact Indramat Customer Service if you do not have a copy of this Guide.*

1.2 Manual Overview

Chapter 1 - Introduction	*Describes the CLC's general theory of operation and its motion capabilities.*
Chapter 2 - CLC Input/Output Systems	*Describes the functionality of the built-in, memory-based I/O registers and the CLC's I/O mapper.*
Chapter 3 - Parameters	*Lists the CLC parameters, divided into four functionally related groups: System, Task, Axis and Drive.*
Chapter 4 - VisualMotion Menu Commands	*Describes the menu commands used in VisualMotion™.*
Chapter 5 - Programming Concepts	*Describes the basic programming elements that are common to both the CLC Icon and Text languages and their implementation.*
Chapter 6 - Icon Programming	*Describes how to use icons and other VisualMotion commands to create motion programs.*
Chapter 7 - Text Language Programming	*Describes the CLC's Text Language programming capability, which permits writing CLC programs using Windows' Notepad or a similar ASCII text-only editor.*
Chapter 8 - Programming Debugging	*Discusses debugging strategies and describes the CLC compiler base code. It also provides a listing of Icon and Text language compiler error messages, as well as various CLC status and diagnostic messages.*
Appendix A - Direct ASCII Communications	*Describes the CLC serial communication protocol, which can be used to communicate with a wide variety of devices and programs.*
Appendix B - DDE Client Interfaces	*Includes examples of how to create custom DDE client interfaces for the CLC card using Excel and Wonderware*
Appendix C - ELS Configuration	*Describes the CLC / Drive resident cams and how to build, run and modify cam applications.*
Appendix D - CLC Drive Parameter Editor	*Describes the CLC Drive Parameter Editor which includes drive tuning, limits and mechanical setup, as well as oscilloscope and diagnostic applications.*

1.3 Overview of CLC

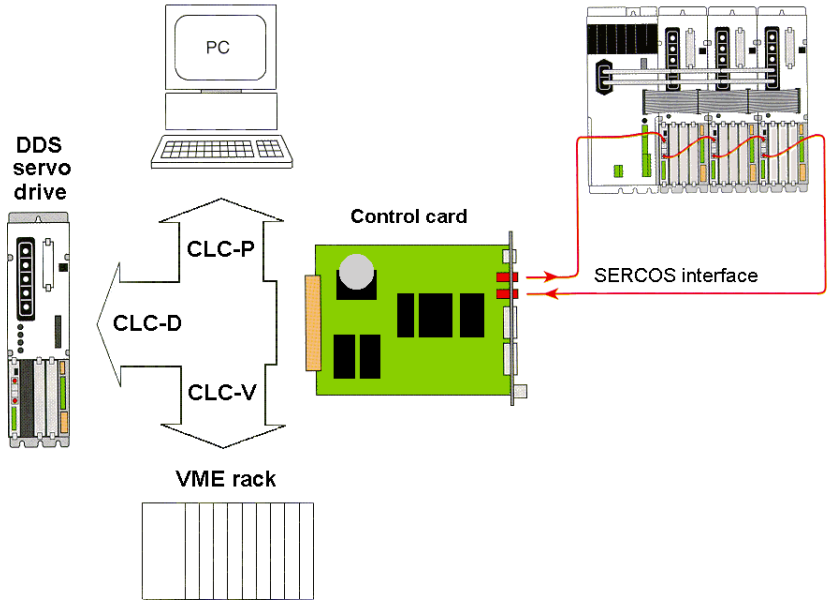
CLC System Architecture

The CLC card is a part of a larger motion control system which also includes digital servo drives and SERCOS, a fiber-optic communication system. The CLC can provide multi-axis coordinated and non-coordinated motion control with tightly integrated I/O logic control functions. The flexibility of the CLC allows it to be used for a wide variety of applications, from general motion control to sophisticated multi-axis electronic line shafting (ELS) to robotics.

CLC controls use SERCOS (SErial Realtime COmmunications System) fiber-optic interface to interconnect with Indramat drives. The SERCOS interface is an internationally accepted standard for real-time high-speed digital communication. It requires only a single daisy-chained fiber-optic cable interconnecting the drives with the control. Synchronous data protocol guarantees response time. It provides continuous monitoring and diagnostics for all devices and includes comprehensive standardized definitions for control loop functions.

The CLC's open bus architecture provides easy I/O interfacing to virtually any manufacturer's PLC's and I/O cards. Both digital and analog I/O are available. Interrupt-type inputs can be used to provide minimum response time recognition of external events.

Four versions of the CLC card are currently available:



CLC-V

VME bus architecture.

CLC-P

IBM PC-AT bus architecture.

CLC-P02

PC104 bus architecture.

CLC-D

Plugs into Indramat's digital drives, providing an exceptional cost-effective motion control solution.

Indramat's VisualMotion® Programming Interface

The CLC motion control card combines an integrated multi-tasking environment with a unique graphical Windows based programming interface - VisualMotion® (VM). VM provides simplified point-and-click programming, operation and management. With this software, system builders have a flexible and comprehensive environment, including easy DDE integration with applications such as Wonderware, InTouch or Visual Basic programs.

CLC BTC06

An optional BTC06 can also be used to control CLC operation and adjust

CLC parameters. The BTC06 is a hand-held instrument with 16 x 40 character display and a 48-key sealed membrane keypad. The pendant provides a convenient operation and position programming interface for Indramat CLC Motion Control. **Refer to the *CLC Start Up Guide* for more information.**

CLC Operating System

A CLC card can simultaneously control up to four independent user tasks (A, B, C & D). Each task can control a coordinated group of two or three axes and any number of independent motion axes. Depending on the application, a single CLC may control up to 40 axes.

CLC multitasking is cooperative and can consecutively execute one instruction from each task. Special event functions preempt normal multitasking for real-time response and are prioritized according to the hierarchy of the event's associated task (Task A - highest priority, Task D - lowest priority). Two additional executive tasks manage the user interfaces through serial communication ports. One controls communication to the Host PC, the other controls the BTC06 when used.

Each task also has a queue for events, permitting asynchronous initiation of multiple independent events within each task. CLC events are a privileged form of subroutine, suspending all CLC program tasks until pending events are done. Each task may have up to four events active at one time. The CLC provides several event types including time-based, distance-based, and position-based events, and others.

Task A events have the highest priority, while events associated with Task D have the lowest. Therefore, an event currently being executed within Task C will be suspended by an event from Task B. Once the active events are completed the suspended program tasks automatically re-activate and resume execution.

1.4 CLC Motion Capabilities

The CLC supports three kinds of motion (non-coordinated, coordinated, and electronic line shaft) and several modes of each type.

Non-Coordinated Motion

Non-coordinated motion is primarily used to control a single independent axis. There are two modes of non-coordinated motion: Single Axis, for linear positioning required to achieve point-to-point movement, and Velocity Mode, as might be used for some spindle motor drives.

Single axis

Single axis motion commands within a user program are interpreted by the CLC and sent to the DDS drive. The user program tells the DDS drive the speed, and/or distance and acceleration it should use to internally develop a velocity profile, which is then maintained and controlled within the intelligent DDS drive. Consequently, single axis motion does not require continuous calculation by the CLC and consumes minimum CLC resources.

Velocity Mode

Velocity mode controls only the speed of the axis, without any position information. The intelligent DDS-2 drive maintains torque and velocity loops internally, updating the internal loops every 250 microseconds.

Ratioed Axes

A special form of non-coordinated motion permits linking two axes by relating the number of revolutions of a slave axis to a master axis. For example, a ratio might be required when the positioning axis of a gantry robot, having a motor on each side of its supporting track, must travel along a circular track.

Coordinated Motion

The CLC defines multi-axis coordinated motion in terms of a path composed of standard straight line and circular geometry segments. Point positions, (x, y, z), are used to establish the start, middle or end of a geometry segment. Two points define a line, three points define a circle. The path combines these standard geometry segments so that the start of the next segment begins at the end of the previous segment. A path, therefore, is nothing more than a collection of connected segments.

Since each segment has an end point specifying speed, acceleration, deceleration and jerk, each segment can have a unique rate profile curve. A special type of segment, called a blend segment, can be used to join two standard geometry segments. Blend segments provide the capability of continuous smooth motion from one standard segment to another without

stopping. They reduce calculation cycle time as well as provide a means of optimal path shaping.

The CLC is capable of calculating a path in any of several different modes:

Constant Speed

Constant Speed mode is always active and tries to maintain a constant speed between any two connecting segments in the path. This mode is constrained by the system's acceleration and deceleration. Constant speed is the optimum path motion for applying adhesives or paint, and welding and some forms of cutting such as laser or water-jet, etc.

Linear Interpolation

A coordinated motion straight line segment is defined by two points. The motion is calculated from the end point of the last segment, or the current position if the system is not in motion, to the new end point. Multi-axis coordinated motion is used when a relationship must be maintained between two or more axes during motion.

Circular Interpolation

A coordinated motion circular segment is defined by three points. Circular motion begins with the end point of the last segment executed, or the current system position if the system is not in motion, moves in a circular arc through an intermediate point, and terminates at the specified endpoint.

Kinematics

In addition to the standard linear and circular segments, the CLC has the capability of executing forward and inverse kinematic movement by using an application-specific library of kinematic functions. Kinematics must be developed by Indramat to customer specifications. Contact Indramat Applications Engineering to inquire about applications which could benefit from kinematics.

Electronic Line Shaft (ELS)

An Electronic Line Shaft is used to synchronize one or more slave axes to a master axis. An ELS master can be a real or virtual axis. A real master can be another axis in the system, or an external feedback device such as an encoder. A virtual master is a command generated by the CLC. *(See ELS Icon, Chapter 6)* Each slave axis can use either **velocity**, **phase** or **cam synchronization**. An ELS also includes the capability to jog each axis synchronously or independently, and to adjust phase offset and velocity while the program is running.

Velocity synchronization relates slave axes to a master in terms of rotational rate. It is used when axis velocities are most critical, as in paper processing operations in which two or more motors act on a single piece of fragile material.

Phase synchronization maintains the same relative position among axes,

but adjusts the lead or lag of the slaves to the master in terms of degrees. It is used when the positions of axes are most critical. For example, to achieve proper registration in printing operations, the axis controlling the print head may be programmed for a particular phase offset relative to some locating device, such as a proximity switch.

Cam synchronization is used when custom position, velocity or acceleration profiles are needed at a slave axis. These special profiles are developed at the slave by sending position commands every SERCOS cycle.

A cam is an (x, y) table of positions that relate a master axis to a slave. Cams can be stored on the CLC or on the digital drive. CLC cams have more adjustment options and can work with any SERCOS drive. Drive cams are more efficient and can be applied to more axes. The same programming commands and utilities are used for both drive-resident and CLC-resident cams.

See Appendix C - ELS Configuration for more information.

2 CLC Input/Output Systems

2.1 I/O Overview

CLC I/O (input/output) is used for operating functions such as System, Task and Axis control. I/O is also used to track system status and diagnostics. The CLC Executive, VisualMotion Program Instructions and remote I/O systems can all read and write I/O data.

The CLC uses an internal block of memory to manage I/O systems. The memory is arranged in a linear array of “Registers” which are 16 bits wide. Each bit can have a value of 1 or 0 which corresponds with its “On” or “Off” state. The CLC-V memory allocation allows for a maximum of 1024 I/O registers, while the CLC-D and CLC-P allow for a maximum of 512.

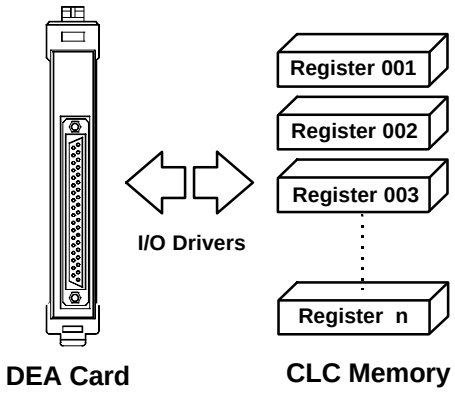
The following registers are reserved by the CLC for system functions:

Register 1: System Control
Registers 2-5: Task Control
Register 6: System Diagnostic Code
Registers 7-10: Task Jog Control
Registers 11-18, 209-240: Axis Control
Register 21: System Status
Registers 22-25: Task Status
Registers 27 and 28: Eagle Module Inputs/Outputs (CLC-V Only)
Register 29: ELS Master Control Register
Register 30: ELS Master Status Register
Registers 31-38, 309-340: Axis Status
Registers 40-87: DEA (4/5/6) I/O
Registers 88,89:
Registers 90 and 91: Latch and Unlatch
Registers 95-97: BTC06 Teach Pendant Status Registers
Registers 98, 99: Teach Pendant Control - Task A-B, C-D
Register 400-405, 410-415: DEA (28/29/30) I/O Registers

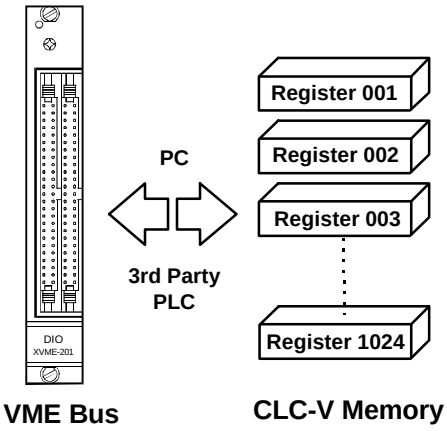
The remaining registers can be used for remote I/O systems and can function like the registers used in any programmable controller.

2.2 Remote I/O Systems

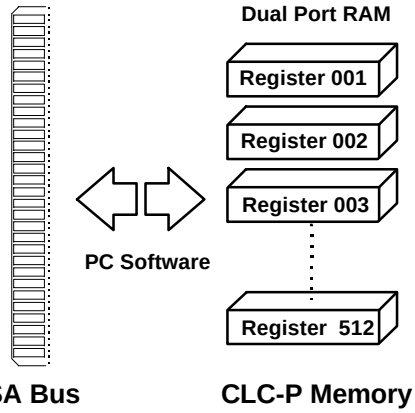
There are several hardware configurations which allow the CLC to read from and write to remote I/O systems. Remote I/O can be accessed through DDS-drive based I/O cards, VME based I/O subsystems (CLC-V only), or PC based I/O subsystems (CLC-P only). ***See Chapter 4 - I/O Setup under the Setup Menu for more information***. Remote I/O can also be accessed through the SERCOS loop using a Lutze I/O device. ***See Parameter C-0-2014 for information on SERCOS I/O Stations.***



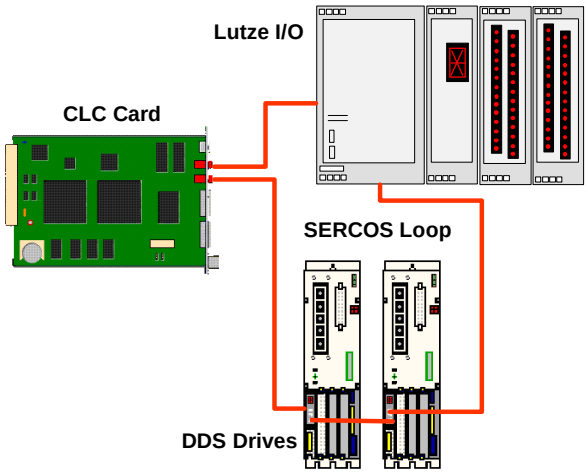
DDS drive based I/O cards: DEA 4.1, 5.1, 6.1
CLC Registers 40-87 are dedicated to this card



CLC-V VME Backplane Communication



CLC-P PC based I/O subsystems



Lutze I/O initialized in the loop at startup SERCOS phase 2

Since the CLC uses industry standard bus architecture, it can control virtually any bus-based I/O system, as long as the specific I/O device software drivers are available. The amount of I/O registers available for a specific CLC system design depends upon the manufacturer, the size of the installed I/O system and the system specific configuration. ***See the***

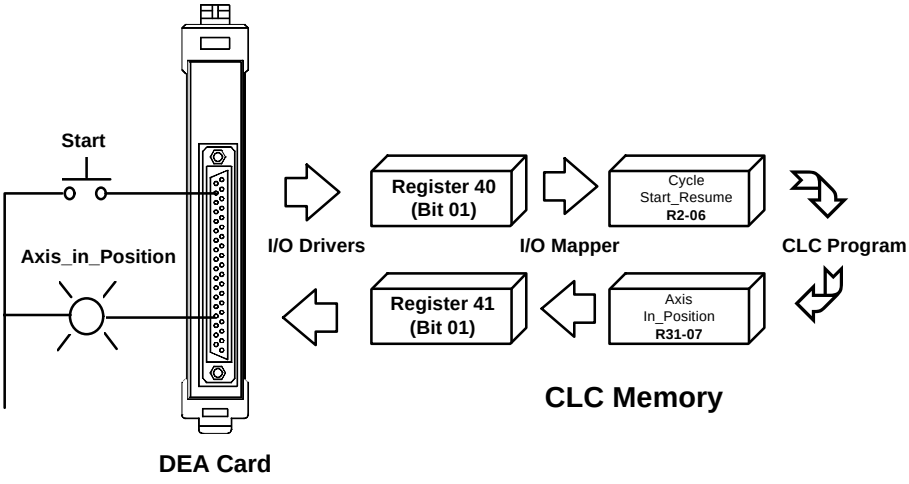
appropriate hardware Appendix in the CLC Start Up Guide - IAE 68010 for selecting I/O subsystems.

In addition to using program I/O instructions to change I/O lines, I/O activity may be related to motion by using high priority motion or time-related "Event" subroutines. Events behave in a manner similar to real-time hardware interrupts, and may be programmed to trigger at a specific distance along a commanded motion or at a specific time related to the motion. ***See Chapter 5. Programming Concepts - Events, for a description of event functions.***

2.3 I/O Mapper

The I/O Mapper is a background function used by the CLC Executive to logically link or “map” the state of bits within the I/O registers. The logical relationship between these bits is defined in a file that contains a list of Boolean equations. These equations can be written in Visual Motion using either the ladder logic editor or the Boolean text editor.

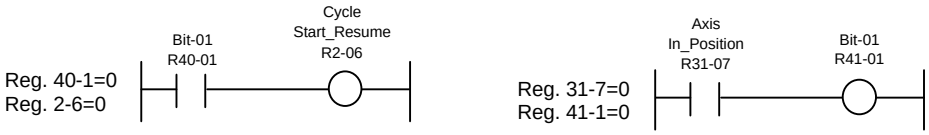
The I/O Mapper allows the user to specify logical system relationships between physical I/O lines and CLC I/O bits. System builders can create custom I/O maps for individual applications that can be saved and transferred to any CLC. During normal operation the I/O Mapper is transparent to user programs. The I/O Mapper equations are executed within 250 µs every 4 ms, regardless of the SERCOS cycle time.



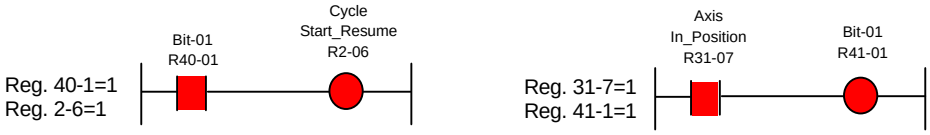
Example: When a physical start switch closes, the DEA card writes to Input Register 40, which is mapped to Register 2 - bit 6 (*Cycle_Start_Resume)*. This initiates an action in the CLC program execution. When Axis 1 reaches its target position, the state of Register 31 bit 7 (*Axis_in_Postition*) is mapped to Output Register 41 bit 1. The DEA card reads this data and a signal lamp is energized.

I/O Mapper Ladder Logic Format

The CLC I/O mapper equations can be represented in a ladder logic format using the VisualMotion Ladder Editor. In this format the previous example would be represented by the following diagrams:



If register 40 bit 1 received input from a physical I/O switch, its state (open or closed) would change the state of Register 2 bit 6 (on or off). This is similar to an electrical switch energizing or de-energizing a coil or a lamp.



For more information on the I/O Mapper and the Ladder Editor see Chapter 4.- I/O Mapper under the Data Menu.

I/O Mapper Boolean Format

The CLC I/O Mapper equations can also be represented using Boolean logic.

(Control_register # - bit #) = (Input_register # - bit #) (Operator) (Input register # - bit #) ...
(Output_register # - bit #) = (Status_register # - bit #); Comments

register: An integer value for an I/O register

bit: An integer value in the range 1 to 16 or a hex mask

Operator*:* One of the I/O Mapper logical operators (listed below)

The left side of the expression is set to the evaluated result of the right side of the expression. The string is parsed left to right. Operators are executed as they are parsed, with all operators having the same level of precedence.

A semi-colon at the end of an I/O Mapper string will allow comments to be entered.

Example:

2-1=100-2; sets Task A, Auto Mode (Register 2, bit 1) to the state of register 100 bit 2.

I/O Mapper Operators

Valid I/O Mapper operators include:

! **NOT**, the complement operator. The bit is inverted.

& **AND**, a bitwise and. If both bits are 1, the result is 1, else the result is 0.

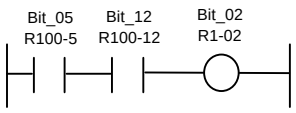
| **OR**, a bitwise inclusive or. If either bit is 1, the result is 1.

( ) Parentheses allow calculation of intermediate results or forcing of precedence. Up to 16 nested operations can be contained in one string.

Example:

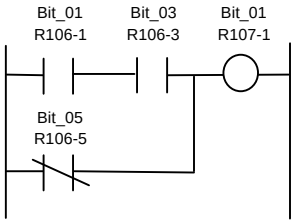
1-2 = 100-5 & 100-12

Sets register 1, bit 2 TRUE if register 100 bit 5 and register 100 bit 12 are both True.



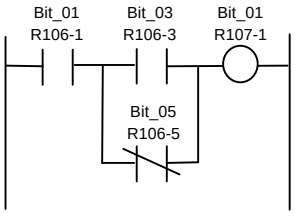
107-1=106-1&106-3|!106-5

Two “And” bits “OR’d” with an inverted bit



107-1=106-1&(106-3|!106-5)

Two “Or” bits with an “And”



The string is processed from left to right if no parentheses are used, The “AND” and “OR” operators have equal precedence.

I/O Mapper Considerations

Directly changing a mapped bit, using VisualMotion or Direct ASCII Communication, may only result in a momentary change of state. The direct change will immediately modify the bit. However, at some time, within the CLC executive cycle time, the I/O Mapper may alter the bit state according to the user specified mapping.

The mapper must complete all operations within 4ms, leaving room for user tasks to run.This includes reading all CLC internal I/O registers, evaluating and mapping all logical expressions, then writing to all physical I/O registers.

The logic strings used to map I/O may also be accessed through the CLC's serial port as a multi-step parameter list (C-0-3000) by using the CLC's Direct ASCII Communication capability. ***See Chapter 3. Parameters for more related I/O Mapper information:***

C-0-3001 I/O Mapper Options C-0-3003 I/O Mapper Total Operations

C-0-3004 I/O Mapper File size C-0-3005 I/O Mapper Executable Size

2.4 I/O Bit Forcing

The CLC provides a register forcing capability that allows a Host system to directly change the state of individual I/O register bits overriding both the physical I/O and the CLC I/O Mapper. Bit forcing circumvents the problem of the I/O Mapper re-mapping a bit that you need to change directly.

Two forcing values allow changing I/O register bits. A 16 bit forcing mask value enables forcing for individual bits in a selected register. A 16 bit forcing state value may then change the state of the enabled individual I/O bits.

The following algorithm mathematically describes how the final register state is set:

New register state = (old state & inverted forcing mask) | (forcing state & forcing mask)

Example:

forcing mask	0000 0000 0011 1000
forcing stat	0000 0000 0000 1000
inverted forcing mask	1111 1111 1100 0111
old register state	0000 1101 1010 0001
	------------------------
new register state	0000 1101 1000 1001

When the forcing state changes bits in a CLC control register, all edge detection is reset. Forcing remains in effect until the forcing mask is cleared to zero. A 1 in a forcing mask and a 0 in the forcing state will force a zero in the new register state.

NOTE: I/O forcing is provided as a system configuration and debugging tool. Forcing can modify control register bits affecting the safe operation of the system. A failure of serial communication between the Host and CLC will prevent the Host from being able to clear a forcing mask, possibly during active motion in the system. USE CAUTION!

For more information on I/O bit forcing see Chapter 4 - Registers under the Data Menu.

2.5 Reading and Writing Physical I/O

The CLC's physical I/O lines are read and written once during a 4ms system scan cycle (2 or 4ms). The fixed scan time provides a deterministic response and I/O sub-system independence from the user tasks.

Within each cycle, the I/O lines are always read before they are written. Near the beginning of the system scan time, the I/O system-specific drivers read data from the I/O lines and store the data to the CLC's I/O register array. The data then remains accessible to user programmed tasks during the CLC's normal multitasking. Towards the end of the system scan time, the I/O drivers transfer the contents of the CLC's I/O registers to the physical I/O registers.

Note that for very high-speed events it is possible to change an I/O register state between these two scans, without affecting the physical I/O line itself.

Although it is possible to read or write to I/O systems directly when the I/O ports reside in CLC accessible memory, this is discouraged. It is advisable to use the CLC's I/O system to access I/O lines. Direct reading and writing of I/O requires a high level of expertise with the CLC, the specific I/O sub-system and the host bus system.

2.6 CLC Registers

The CLC reserves the first 100 registers for system and task, control and status functions. If Indramat drivers for Indramat supported I/O subsystems are used, registers 100 to 119 are typically reserved for inputs and registers 120 to 139 are typically reserved for outputs. A default I/O mapping is used to map physical I/O lines to the appropriate CLC system and task control register bits. However, as previously noted, all CLC physical I/O depends upon the user installation and system configuration.

The following sections describe the CLC reserved registers, the function of control and status bits within these registers, and the I/O mapping provided as a system default mapping.

Register 1: System Control

The first CLC register is reserved for system control. The System Control register bits are dedicated to system supervisory control functions.

Register-Bit	Function
1-1	Parameter Mode
1-3	nEmergency Stop
1-5	Clear All Errors
1-6	Pendant Live Man
1-8	Activate Program
1-9	Program Select LSB
1-10	Program Select Bit 2
1-11	Program Select Bit 3
1-12	Program Select MSB
1-14	Pendant Enable
1-15	Pendant Level LSB Must be set to 1's to enable teach pendant editing and function keys
1-16	Pendant Level MSB Must be set to 1's to enable teach pendant editing and function keys

Bit 1: Parameter Mode/ nRun Mode

When this bit changes from (0) to (1), Parameter Mode is selected. All user tasks are immediately stopped. The system is switched into parameter mode, and the drives are switched into SERCOS phase 2.

When this bit changes from (1) to (0), the system is re-initialized into Run Mode. Parameter initializations are performed and the drives are switched from phase 2 to phase 4. If there are no errors, the user tasks are ready to operate.

Bit 3: nEmergency Stop

This input is active low (0 = Emergency Stop). When it is set to (0), all user tasks are stopped except those selected to run during errors. All motion is stopped, and the drives are set to zero velocity and disabled. When it is set to (1), the emergency stop condition has been corrected, and the tasks can run if there are no other errors

Bit 5: Clear All Errors

When this bit changes from (0) to (1), any existing errors are cleared. This includes system, task, and drive errors. If drives were previously enabled, they are restored to an enabled state. To run the tasks, another cycle start transition is required, unless the automatic start option is enabled. To clear errors, a transition of (0) to (1) is required.

Bit 6: Pendant Live Man

This bit should be wired to the teach pendant live man switch when the teach pendant is used. When it is (0), no motion can be initiated from the teach pendant and any motion in progress is immediately stopped. When it is (1) (live man closed), the teach pendant can jog, start, and stop motion.

Bit 8-12: Activate Program and Binary Program Select

The active program can optionally be changed using I-O bits 9-12 in the System Control Register. A transition from (0) to (1) on Activate Program bit 8 will activate the program based on bits 9-12. These bits correspond to the program number (from 1 to 10). If the Activate Program bit is high at power-up, the program selected with the Binary Program Select bits will be activated.

Note that the user interface (Visual Motion, Teach Pendant, etc.) selections take precedence over the bits. The actual active program is acknowledged in System Status Register bits 9-12.

Example:

To activate program 5, set the bits as follows:

Bit 9: 1

Bit 10: 0

Bit 11: 1

Bit 12: 0

Then Bit 8 requires a transition from 0 to 1.

Bit 14: Pendant Enable

This bit toggles control of tasks and jogging between the teach pendant and the I-O system. When it is set to (0), the system I-O mapper and control registers are in control. When it is set to (1), the teach pendant assumes control of system functions, forcing all relevant bits in the control registers

Bits 15-16: Pendant Access Level

These bits provide access protection for teach pendant menus. The protection levels are defined per-menu to provide restricted access to data. If a menu's protection level exceeds the value of these bits, the menu can be viewed but not edited.

Registers 2-5: Task Control

The CLC reserved registers 2, 3, 4 and 5 are dedicated to task control for Tasks A, B, C and D respectively. The state of the control register bits determine how the CLC executive executes the user task programs. User programs may read, set or clear bits in these registers to monitor and control each task's status.

The task control bits permit independent control of the operations within each of the four tasks in a user program.

Setting Auto Mode enables execution of a task. The current version of the software restarts the task at the beginning of the program. Future revisions may provide a more flexible restart method.

Clearing Auto Mode switches the CLC into Manual, ramping down all axes used by a task and aborting the task without finishing the current instructions.

Cycle Start/Resume starts or resumes the selected task.

A Cycle Stop finishes current instruction. An immediate Cycle start resumes the program at the next instruction.

Single Step executes one instruction at a time.

Bit	Function
1	Mode:Auto /nManual
2	Override Automatic Start
4	Single Step
6	Cycle Start /Resume
7	nTask Stop
9	Task Event Trigger
10	Trace Enable
11	Breakpoint Enable
12	Sequencer Single Step
13	Step Sequence Function

Registers 2 - 5 for Tasks A - D

Bit 1: Mode: Automatic/ nManual

This bit selects the mode of operation for a task. When it is (0), the task is in Manual Mode, the user program does not run, and manual jogging is enabled. When it is (1), the task is in Automatic Mode and is ready to begin execution.

Switch from Manual to Automatic (0 to 1):

The instruction pointer is reset to the beginning of the program, and all events become inactive. At the next (0-1) transition of the cycle start bit when the cycle stop bit is (1), the program starts running.

Switch from Automatic to Manual (1 to 0)

The user task and any events associated with it stop execution immediately. Any motion associated with the task is immediately decelerated to zero velocity. Coordinated, single-axis, and velocity axes are stopped using the maximum deceleration. The ELS master is stopped using the E-Stop deceleration. The instruction pointer is set to where the program was stopped, but is reset to the beginning of the program when the task is returned to Auto Mode.

Bit 2: Override Automatic Start

The 'Automatically Start Task' parameter option (T-0-0002, bit 4) allows a task to start immediately upon exit from parameter mode or clearing of an error. The "Override Automatic Start" bit can be used to temporarily disable this function. The bits in the task control register are enabled as long as bit 2 is high (1), and are ignored when the bit is low (0). This allows the automatic start option to be disabled in case the task needs to be stopped or the debugging bits need to be used.

Bit 4: Single Step Select

When Bit 4 = 1, the task is placed in Single Step Mode. Each positive (0 to 1) transition of Cycle Start bit executes one user task program instruction then pauses (providing that the system is in Automatic mode and nTask Stop is inactive). If this bit is set while a task is running the current instruction completes. The task then pauses and waits for a Cycle Start transition.

Event functions cannot be single-stepped. If one or more events have been started or queued by executing the current instruction the events will always continue to completion before pausing the user task program.

When Bit 4 = 0, the task is in Normal Run Mode. When Single Step is zero, normal cycling begins at the next positive transition of the Cycle Start bit (providing that automatic mode is true and nTask Stop is false).

Bit 6: Cycle Start/Resume

When this bit is changed from (0) to (1), the user task starts executing at the current instruction, if the task is in automatic mode, the nTask Stop bit is (1), and there are no errors. It is also used to resume the task after a task stop and to restart the task after entering automatic mode. If single-stepping is enabled, the next instruction is executed with each positive transition.

A (0 to 1) transition is required to start or resume the task

Bit 7: nTask Stop

When Bit 7 = 0, the **n**Task is stopped. A negative transition (1 to 0) stops the task program at the end of the current instruction. All types of motion are halted and can be resumed at the next Cycle Start. The **n**Task Stop bits function as a “Pause”. When the bit is set to 0, the CLC pauses execution of the task and the instruction pointer remains at the current instruction. When the bit is again = 1 and the cycle start bit is toggled, the task resume execution at the current instruction.

All types of events will execute during a cycle stop state. Only the main program flow in tasks A, B, C and D are affected.

A **n**Task Stop decelerates **Coordinated Motion** to zero velocity after the current instruction. Motion is then paused on the current segment. All distance and time-based events remain active. As long as the task is in automatic mode, the next 0 to 1 transition on the cycle start will resume motion and complete all pending segments.

Single axis motion halts each axis in the task by decelerating the axes to zero velocity while retaining target position. The previous state of the GO command is saved until the next cycle start. If the GO command was active, motion will be resumed at the next cycle start in automatic mode. All normal and repeating events remain active.

All **Velocity** mode axes are decelerated to zero velocity if ramping is enabled, or set to zero velocity if step command is selected. All events remain active.

The **ELS** master axis is decelerated to zero velocity. The previous state of the GO command is saved until the next cycle start. If the GO command was active, the master will be commanded the last programmed velocity. All events on the ELS slaves remain active. The slave axes remain synchronized to the master if synchronization was enabled. Because the CLC has no control over a real master, motion of the slaves of a real master cannot be changed.

Ratioed slave axes always follow the master axis. Events remain active and motion will continue until the master axis is stopped.

The operation of **Torque** mode axes during a **n**Task Stop are not defined at this time.

NOTE: *nTask Stop does nothing to assure that the system is in a safe or known condition to stop. nTask Stop simply completes the current instruction, then ramps down motion in the task.* ***See "Cycle Stop in User Programs" in CLC Cycle Control Considerations, for more information.***

Bit 9: Task Event Trigger

This bit is reserved as an Event Interrupt Input for each task. Each low-to-high (0 to 1) transition of this input will trigger an event to the corresponding task. This event type can be used to start a process or to respond to an external event.

In the event table, Type 6 selects an Interrupt Input event. The event/trigger

(arm event) instruction enables the interrupt input. The event/done (disarm event) instruction is used to disable the input. The CLC scans the input every 4ms and queues an event upon a low-to-high transition. The event function will take priority over the user tasks, allowing quick response to an external input.

The I-O mapper can be used to reverse the logic of the interrupt input, or to direct other external inputs to it. Logic in the event function can then scan the multiple inputs to determine the source of the interrupt.

Bit 11: Breakpoint Enable

When this bit is set to (1), the breakpoint enabled in task parameter Tx.137 is active. When program flow reaches the breakpoint, the task is stopped. When this bit is set to (0), the program executes normally, without breakpoints.

Bit 12: Sequencer Single Step

This bit places the CLC into Sequence Single Step mode. As long as this bit is (1), a (0-1) transition on the cycle start bit causes the program to be stopped after each sequencer step is executed. If this bit is (0), the sequencer executes normally.

Bit 13: Step Sequence Function

This bit places the CLC into Function Single Step mode. As long as this bit is (1), a (0-1) transition on the cycle start bit causes the program to be stopped after each sequencer function is executed. If this bit is (0), the sequencer executes normally.

CLC Cycle Control Considerations

Cycle Stop in User Program

A cycle stop implies that a task's motion cycle has completed and the system is at a safe place to halt. The nTask Stop bit cannot always be used for this purpose, since it only stops task instruction execution and commands the drives to ramp down.

A nTask Stop signal may lose track of user task activity that is related to axis or segment position or time. In addition, the CLC's path planner may have several queued segments or events. Queued events always continue execution until completion. This may result in the system position and I/O losing synchronization with your programmed sequence of task instructions. If you attempt to simply restart motion the results may not be predictable.

If each of your tasks require a cycle stop capability, a separate user task I/O bit should be user configured into the I/O system for each task needing a cycle stop. Your program then tests the associated I/O bit from within your task program. Use the condition of the I/O bit to branch to a program routine that halts motion and establishes a known system state. Since you are programming a unique system, only you can determine a safe system condition.

System Parameter Mode

A switch to Parameter Mode immediately disables all user tasks, and switches the DDS drives to SERCOS Phase 2.

System Shutdown Errors

A shutdown error disables all drives, stops coordinated and single-axis motion, and puts the task into manual mode. All control bits are left at their current state.

Programmed End of Task

A task program that reaches and executes the task/end instruction or the Finish Icon has the same effect as activating the *n*Task Stop I/O line.

Register 6: System Diagnostic Code

This status register shows the current CLC diagnostics code, in Motorola 16-bit format.

Registers 7-10: Task Jog Control

The reserved registers 7, 8, 9 and 10 are dedicated to jogging control for Tasks A, B, C and D respectively.

Axis Jogging

Any drive-controlled axis (single-axis, velocity, or ELS) may be jogged independently when its associated task is in manual mode. ELS axes can also be jogged with the master. ***See Chapter 4 - Coordinated Motion under the Setup Menu***.

Coordinated Jogging

Coordinated axes may be jogged in any direction while a task is in manual mode with the Task Jog Register. The coordinate to jog, type of jog, and parameters to use are selected using bits in this register. Single-axis and velocity mode axes are jogged using the Axis Control Registers.

Motion is started with a low-to-high transition on either the jog forward or the jog reverse bit. Motion is stopped when both jog bits are low, when both jog bits are high, when the task mode selection changes, or when a travel limit or incremental distance has been reached.

Jog mode (continuous or incremental) and jog speed or distance are selected in the Task Jog Register for the task associated with the axis.

Register-Bit	Task	Function
7-1	A	Continuous/Incremental
7-2		Coordinated Jog Forward
7-3		Coordinated Jog Reverse
7-4		Jog Type LSB
7-5		Jog Type MSB
7-6		Distance/Speed
7-9		Jog X Coord
7-10		Jog Y Coord
7-11		Jog Z Coordinate
7-12		Jog Joint 4
7-13		Jog Joint 5
7-14		Jog Joint 6
8-1 through 14	B	Same as Task A
9-1 through 14	C	Same as Task A
10-1 through 14	D	Same as Task A

Bit 1: Mode: Continuous/nStep

This bit is used to select the jog mode for both coordinated and single-axis jogging. The mode takes effect when the next jog is started with a transition on the jog forward or jog reverse bit.

Bit 1=0, nStep jogging. Motion stops after the large or small distance is reached, or when the jog bit is set to 0.

Bit 1=1, continuous jogging. Motion stops when the travel limit is reached on an axis or when the jog bit is set to 0.

Bit 2 and 3: Coordinated Jog Forward (bit 2) and Reverse (bit 3)

Coordinated Motion: A low-to-high (0-1) transition on this bit while a task is in manual mode causes motion to start in the positive (Bit 2) or negative (Bit 3) direction in the coordinate selected in bits 9 to 14. A high-to-low (1-0) transition immediately stops the motion.

ELS Motion: A low-to-high (0-1) transition on this bit while a task is in manual mode causes motion to start on the ELS master in the positive (Bit 2) or negative (Bit 3) direction. All axes with the enable synchronized jog bit set in the axis control register will follow the master. A high-to-low (1-0) transition immediately stops the motion.

Bits 4 and 5: Jog Type

Bits 4 and 5 select the type of coordinated jogging for the next jog motion.

World Jog: jogs the axes in the world coordinate selected by bits 9, 10, and 11.

Joint jog: jogs individual axes, with the joint number determined in bits 9 through 14.

Tool jog: jog axes in the tool coordinates selected with bits 9, 10, and 11. It is available only in future 6-axis robot versions.

Bit 5	Bit 4	Jog Type Selected
0	0	World Jog
0	1	Joint Jog
1	0	Tool Jog (6-axis robot version)
1	1	not used

Selecting an invalid jog type will issue a warning message.

Bit 6: Distance/Speed (Large/nSmall, Fast/nSlow)

When bit 6 is set to 0 before a continuous jog, the slow jog speed is selected. For an incremental jog, the small distance and slow speed are selected.

When bit 6 is set to 1 before a continuous jog, the fast jog speed is selected. For an incremental jog, the large distance and fast speed are selected.

Bits 9 through 14 select the coordinate or joint that will be jogged when the jog forward or jog reverse bits are activated. Only one of these bits should be set, since jogging is allowed in only one coordinate at a time.

Bit number	World and Tool Jog	Joint Jog
9	X Coordinate	Joint 1
10	Y Coordinate	Joint 2
11	Z Coordinate	Joint 3
12	Roll Axis (SAR only)	Joint 4 (SAR only)
13	Pitch Axis (SAR only)	Joint 5 (SAR only)
14	Yaw Axis (SAR only)	Joint 6 (SAR only)

When jogging in world coordinates, motion will be generated parallel to the selected X, Y, or Z coordinate according to bits 9 through 11. For example, setting bit 9 high and bits 10 and 11 low enables jogging parallel to the X axis.

For jogging of individual joints, the axis to jog is selected in bits 9 through 14. For example, if axes 2, 3, and 4 are used in coordinated motion, bit 9 selects axis 2, and bit 11 selects axis 4. The jog speed used is a percent of maximum axis velocity.

If an invalid jog type is selected, a warning message is issued.

Registers 11-18, 209-240: Axis Control

Registers 11-18 are for Axes 1-8. Registers 209-240 for Axes 9-40. These Registers are reserved for CLC axis control. These axes correspond to the DDS drive positions on the SERCOS fiber-optic communication loop. Any drive-controlled axis (single-axis, velocity, or ELS) may be jogged independently when its associated task is in manual mode. ELS axes can also be jogged with the master. ***See Chapter 4 - Coordinated Motion under the Setup Menu***.

Jog mode (continuous or incremental) and jog speed or distance must be selected in the Task Jog Register for the task associated with the axis. Motion starts with a low-to-high transition on either the jog forward or the jog reverse bit. Motion stops when both jog bits are low, when both jog bits are high, when the task mode selection changes (e.g., from manual to automatic mode), or when a travel limit or incremental distance has been reached.

Bit	Axis	Function
1	1	Disable Axis
2		Jog Forward
3		Jog Reverse
4		Synchronized Jog

Bit 1: Disable Axis

When this bit is set to (1), all motion is disabled for this axis. The drive is immediately set to zero velocity and the position loop is disabled. Motion commands in the user program and from the control have no effect. Note that the drive can produce torque to hold position. It is still in the SERCOS ring. I-O and some diagnostics are still enabled.

When it is set to (0), motion is enabled, provided other conditions allow it and there are no errors on the drive or the control.

Bit 2 and 3: Jog Forward (bit 2) and Reverse (bit 3)

A low-to-high (0-1) transition on this bit while an axis is in manual mode causes motion to start in the positive (bit 2) or negative (bit 3) direction. A high-to-low (1-0) transition immediately stops the motion.

Motion is stopped when both jog bits are low, when both jog bits are high, when the task mode selection changes, or when a travel limit or incremental distance has been reached. Jog mode (continuous or incremental) and jog speed or distance are selected in the Task Jog Register for the task associated with the axis.

Bit 4: Synchronized Jog

ELS slave axes may be jogged independently or synchronously when their associated task is in manual mode.

When the Synchronized Jog bit is set to (0), the axis can be jogged independently. It does not follow the master, and it may be jogged continuously or incrementally in single-axis mode. Axes can be jogged individually to set up phase offsets or to prepare the machine for operation.

When the Synchronized Jog bit is set to (1), the slave axis follows the real or virtual master. This jog mode can be used to initially thread the material into sections of the machine, maintaining the position or velocity relation to the master. The master is jogged using the coordinated jog bits in the Task Jog register.

Register 21: System Status

The System Status register at CLC register address 21, is a read-only register dedicated to system status. The status is indicated by a high level in the appropriate bit.

Register-Bit	Function
21-1	Parameter Mode
21-4	Service Chan Ready
21-5	Error
21-9	Active Program LSB
21-10	Active Program Bit 2
21-11	Active Program Bit 3
21-12	Active Program MSB
21-13	TP Password Active
21-14	Teach Pendant

Bit 1: Parameter Mode/ Initializing

0 = Run Mode, 1 = Parameter Mode or Initializing System

If this bit is (1), the CLC is in parameter mode or the system is being initialized into run mode. If parameter mode is selected in System Control bit 1, the drives are in phase 2, access to restricted parameters is allowed, and the user programs are stopped.

If this bit is (0), the CLC is in run mode, and the user program is ready for operation if there are no errors.

Bit 4: Service Channel Ready

This bit can be checked by a user interface before initiating communication with a drive. When it is (0), the SERCOS ring is disconnected or phases are being switched. When it is (1), the drives are ready for service channel communication.

Bit 5: Error

This is the global error indicator for the CLC system. If the system, any task, or any drive has an error, this bit is set to (1). If there are no errors present, it is set to (0).

Bits 9-12: Active Program

These bits indicate the currently active program as a binary program number. (Bit 12= most significant bit)

Example: The bits below indicate that program 5 is active.

Bit 9: 1 Bit 10: 0

Bit 11: 1 Bit 12: 0

Bit 13: Teach Pendant Password Active

This bit is set if the teach pendant password is active. It allows the user program or I-O mapper to disable functions while the corresponding functions are disabled in the teach pendant.

Bit 14: Teach Pendant Connected

This bit is set if the teach pendant is connected. On the CLC-V, the CTS line on the RS-232/422 connector is indicated in this bit. On the CLC-D and CLC-P, the ZT line on the cable is indicated. These lines are wired by default so that this bit indicates a connection.

Registers 22-25: Task Status

The reserved read-only registers 22, 23, 24 and 25 are dedicated to task status for Tasks A, B, C and D respectively. The condition of the task status registers may be read by a user task program to determine the state of the system and the task's program execution.

Register-Bit	Task	Function
22-1	A	Mode: Auto/Manual
22-2		Coordinated Running
22-4		Single Step
22-5		Task Error
22-6		Task Running
22-10		Trace Ready
22-11		Breakpoint Reached
23-1 through 6	B	Same as Task A
24-1 through 6	C	Same as Task A
25-1 through 6	D	Same as Task A

Bit 1: Mode: Automatic/Manual

This bit is set to (1) by the CLC when the task is in automatic mode and is ready for the program to be started. It is set to (0) when manual mode is selected, an error is preventing the task from starting, or the task has not been initialized into automatic mode.

Bit 2: Coordinated Running

This bit is set to (1) by the CLC when coordinated motion is ongoing in the task. When motion stops, the bit is reset to 0.

Bit 4: Single Stepping

When this bit is set to (1), the program flow has been stopped after an instruction, sequencer function, or sequencer step in single-step mode. To resume program flow, a (0-1) transition on the cycle start bit is required.

Bit 5: Task Error

This bit is set to (1) by the CLC when a task has an error or warning condition. An error is shown in parameter T-0-0122 . The bit is (0) when no errors exist in this task.

Bit 6: Task Running

This bit is set to (1) by the CLC when the user task is executing program instructions, either from the main task or from an event. It is (0) when the task is not running

Bit 10: Trace Ready

Bit 11: Breakpoint Reached

When this bit is set to (1), the breakpoint enabled in task parameter T-0-0137 and activated with Task Control Register bit 11 has been reached. Program flow has been stopped. To resume program flow, a (0-1) transition on the cycle start bit is required.

Registers 27 and 28: Eagle Module Inputs/Outputs (CLC-V Only)

Register 27 on CLC-V

These 12 inputs reside on the Eagle Module of the CLC-V 2.3 through the 25-pin connector on the front panel. They can be mapped to system bits with the I-O mapper or can be read in the user program.

Bits 1-8: Inputs1-8

Bit 9-12: Reserved for future functions

Bits 13-16 Always set to (0)

Register 28 on CLC-V

These four outputs reside on the Eagle Module of the CLC-V 2.3 through the 25-pin connector on the front panel. They can be mapped to system bits with the I-O mapper or can be written by the user program.

Bits 1-4: Outputs1-4

Bits 9-16 Available as flags (not connected to I-O)

Register 29: ELS Master Control Register

This register is functional only in CLC versions GPx-02.10 and greater.

Bit 4: Real Master: Primary/!Secondary

This bit allows the ELS master to be switched between two real masters. The changeover can be done while the master is running.

When it is set to (0), the ELS and cam slave axes follow the primary real master selected in the user program and in parameter C-0-1000.

When it is set to (1), the ELS and cam slave axes follow the primary real master selected in the user program and in parameter C-0-1001.

Conditions for Master Switching:

When the master is switched between primary and secondary, the switch is not completed until the position of the two masters are within the ELS master switching threshold.

NOTE:

- This bit has no effect when no real master is configured in the program or parameters.
- The master switch is acknowledged by the CLC via the real master bit in the ELS status register.

Bit 5: Master Type: Virtual/Real

This bit allows the ELS master to be switched between a real master and a virtual master.

When it is set to (0), the ELS and cam slave axes follow the real master that is selected in the user program or through system parameters C-0-0151 and C-0-0164.

When it is set to (1), the ELS and cam slave axes follow the virtual master. The master may then be stopped, started, or jogged without regard to the real master's motion.

Conditions for Master Switching:

The virtual master can be dynamically synchronized to the real master using a ramp. *For more information refer to the Dynamic Synchronization section in **Appendix C. ELS Configuration.***

When the master is switched from real to virtual, the switch immediately takes place, and the virtual master position is set to that of the real master.

NOTE:

- This bit has no effect when no real master is configured in the program or parameters.
- The master switch is acknowledged by the CLC via the master type bit in the ELS status register.
- The virtual master can be initialized with a velocity and/or a stop position before the master is switched. This allows a controlled stop to be performed on the slave axes by toggling this control bit.

Using the I-O Mapper, this control bit can be tied to the task Auto/Manual bit or Cycle Stop bit, so that the user program can be stopped and started while the real master is running.

Register 30: ELS Master Status Register

This register is functional only in CLC versions GPx-02.10 and greater.

Bit 4: Real Master: Primary/nSecondary

This status bit acknowledges the real master in the ELS master control register. When it is (0), the primary real master is active. When it is (1), the secondary master is active.

If the state of this bit does not match the corresponding bit in the control register, the master switch is still in progress. When switching to a real master, the master positions must be equal before the new master is activated.

Bit 5: Master Type: Virtual/nReal

This status bit acknowledges the master selection in the ELS master control register. When it is (0), the real master is active. When it is (1), the virtual master is active.

If the state of this bit does not match the corresponding bit in the control register, the master switch is still in progress. When switching to a real master, the master positions must be equal before the new master is activated.

Bit 6: Virtual Master Enabled

This bit is set to (1) whenever the ELS virtual master is ready to accept motion commands. In the current software version, the virtual master is always enabled in any ELS mode.

Bit 7: Virtual Master at Programmed Speed

This bit is set to (1) when the current virtual master speed is equal to the programmed velocity (C-0-0153 = C-0-0158). Otherwise, it is set to (0).

Bit 9: Master Direction: Positive/ nNegative

When the currently active master's speed is greater than or equal to 0, this bit is set to (0). Otherwise, it is set to (1).

Bit 10: Master Stopped

When this bit is (1), the ELS master is stopped. When it is (0), the master is running.

It is set to (1) when the absolute value of the active master's current velocity is less than the master zero velocity window: (|C-0-0158 | < C-0-0159).

Bit 11, 12: Position Difference Monitoring

When bit 11 is (1), master 1> (master 2 + position window)

When bit 12 is (1), master 2>(master 1 + position window)

When bit 11 and bit 12 are off, both masters are in sync.

See Parameter C-0-1018 Position Monitoring Window

ELS master position differences can be monitored with parameters and register bits. This ensures an accurate position difference and reduces the processing that needs to be done in the user program.

See Parameter C-0-1017 Position Difference Selection, C-0-1019 Position Difference.

Registers 31-38, 309-340: Axis Status

Registers 31-38 are for Axes 1-8. Registers 309-340 (read only) for Axes 9-40. These registers are reserved for CLC axis status. These axes correspond to the DDS drive positions on the SERCOS fiber-optic communication loop.

Bit	Axis	Function
2	(1-40)	Jogging Forward
3		Jogging Reverse
4		Phase Adjusted
5		ELS Enabled
6		ELS Secondary Mode
7		Axis In-Position
8		Axis Aligned
10		Axis Stopped
11		Axis Halted
12		Class 3 Status
13		Class 2 Warning Change
14		Shutdown Error
15		Drive Ready LSB
16		Drive Ready MSB

Bit 2 and Bit 3: Jogging Forward (Bit 2) and Reverse (Bit 3)

This status bit is set to (1) when the axis is jogging forward (Bit 2) or reverse (Bit 3) through either the teach pendant or the I-O bits. Otherwise, it is set to (0).

Bit 4: Phase Adjusted

For ELS and Cam axes, this bit indicates if a phase adjustment move has been completed. During the phase adjust, it is set to (0), indicating the adjustment is in progress. When the phase adjust is complete, or when first synchronizing, this bit is set to (1).

Bit 5: ELS Enabled

When this bit is set to (1), the axis is in ELS or cam mode. When it is (0), it is in velocity or single-axis mode, as indicated by bit 6.

Bit 6: Secondary Mode

For ELS or cam axes, this indicates the current secondary mode that is enabled when ELS/cam is disabled. If it is set to (0), the axis is in single-axis mode. If it is set to (1), the axis is in velocity mode.

Bit 7: Single Axis in Position

Bit 7 is set to one (1) when the target position is reached and the axis is at zero velocity in single-axis mode. The drive associated with the axis must have its in-position window and zero velocity window parameters set correctly for this bit to function properly.

Bit 8: Axis Aligned (CLC cam axes only)

This bit provides the status of CLC based cam alignment. It is set to (1) if the axis is aligned to the cam, and (0) if it is not aligned. This allows user program logic to determine if an alignment move or phase offset is needed. This bit is only for CLC cam axes. It is not checked for Drive Cams.

The following conditions set this bit to (0):

- The axis is not configured in the program to be a cam axis.
- A valid cam is not active for this axis.
- The absolute value of (position of the axis - slave position from cam equation) is greater than the in-position window (drive parameter S-0-0057).

The following conditions set this bit to (1):

- The axis is synchronized to the master
- The absolute value of (position of the axis - slave position from cam equation) is less than or equal to the in-position window (drive parameter S-0-0057).

Bit 10: Axis Stopped

This bit corresponds to Drive Parameter S-0-0182 bit 1. It is set to (1) when the feedback velocity is less than the drive's zero velocity window.

Bit 11: Axis Halted

Bit 11 is set to one (1) when the drive's restart/nhalt bit is set to zero (0) (drive halted) and the axis is at zero velocity in single-axis mode.

Bit 12: Class 3 Status

This bit corresponds to the Change in Class 3 Diagnostics bit in the drive status word. When the diagnostic condition changes, the drive changes this bit from a (0) to a (1). The program can then check this register bit for a (0) to (1) transition, instead of continually reading the status parameter through the service channel.

Bit 13: Class 2 Warning

This bit indicates that a Class 2 warning condition exists. It is set (1) when a warning occurs and is not cleared (0) until the warning is cleared. Warnings are often temporary conditions. This bit allows the program to latch and take action on a warning.

Bit 14: Drive Shutdown Error
Bit 14 is set to one (1) when there is a Class 1 Diagnostic Shutdown Error in the drive. This bit corresponds to the same bit in the drive's SERCOS status register.

Bit 15, 16: Ready to Operate
Bits 15 and 16 indicate when a drive is ready to operate. When both bits are one (1) the drive will respond to motion commands. These bits correspond to the drive's SERCOS status register. ***See Drive Parameter S-0-0135.***

Bit 16	Bit 15	Description	DDS LED Code
0	0	Drive not ready for power up	error or P1 - P3
0	1	Drive ready for power up	bb
1	0	Drive control and power sections ready	Ab
1	1	Drive ready to operate	AH or AF

Registers 40-87: DEA (4/5/6) I/O

Each DDS can hold up to three DEA cards, each DEA card has 15 inputs and 16 outputs. DEA cards are hardware configured to be card 1, 2, or 3, there order in the drive rack does not matter. DEA cards in drives 1 through 8 have predefined base registers, for cards in drives 9 through 99 a base register must be entered.

Reg #	Name	Reg #	Name
40	DDS1_1IN	46	DDS2_1IN
41	DDS1_1OUT	47	DDS2_1OUT
42	DDS1_2IN	48	DDS2_2IN
43	DDS1_2OUT	49	DDS2_2OUT
44	DDS1_3IN	50	DDS2_3IN
45	DDS1_3OUT	51	DDS2_3OUT
		….87	DDS8_3OUT

Registers 88,89: Task A Extend Event Control

Register 88 (USER_XI_REG) is used to trigger up to 16 events in Task A.

Register 89 (USER_XO_REG) is used to monitor the status of events triggered by Regiser 88.

Together they provide a time critical way to control motion on the CLC card without time wasting polling loops. These events are similar to the Task Input Transition, located in each task control register, but are limited to the task with the highest priority (Task A). ***Refer to the Event Table description in Chapter 5. Programming Concepts for more information.***

Register Operation

At power up and at system reset the output register USER_XO_REG is cleared.

The input register USER_XI_REG triggers an EUI on positive edge bit transitions. The event function assigned to the bit is executed at that time. Only positive going edges in USER_XI_REG can trigger an EUI.

The output register USER_XO_REG is effected by the Event Setup Box Icon arm and disarm functions as well as by bit transitions within USER_XI_REG.

When an EUI is armed through the Event Setup Box Icon its bit is set in USER_XO_REG. This provides an external output that indicates that the EUI is armed and ready for operation. Likewise it is reset if disarmed through the icon.

When an EUI is triggered, its corresponding output bit in USER_XO_REG is reset (normally it is set) providing an external output indicating that the event is active. It remains reset so long as no negative transition on the corresponding input bit in USER_XI_REG detected. When a negative edge

on the input bit is detected in USER_XI_REG, the output bit in USER_XO_REG is again set indicating the event is armed and ready for another positive going edge.

Registers 90 and 91: Latch and Unlatch

Reserved_Registers

Register 90 provides 16 latches that can be set to (1) directly or by the I-O mapper. The bits in register 90 can be reset to (0) only by writing a (1) to the corresponding bit in register 91. Register 91 immediately clears its bits to (0), so that a pulse is not needed on the unlatch bits.

Registers 92-94: Mask Pendant Key Functionality

The bits in registers 92-94 “mask” the functionality of the corresponding bits in registers 95-97. This means that if a bit is set to 1 (on) in register 92, 93 or 94, the BTC06 key controlled by the corresponding bit in register 95, 96 or 97 is not operational. (Register 92 masks register 95, register 93 masks register 96 and register 94 masks register 97.) This masking feature allows the programmer to redefine the action of a key or prevent the action normally mapped to that key.

Examples:

Register 92, bit 1 is set to 1. Pressing the F1 key (status given in register 95, bit 1) does not result in the expected action.

Register 93, bit 7 is set to 1. Pressing the X+ key (status given in register 96, bit 7) does not result in jogging in the Jog Menu.

Note: When a key on the BTC06 is masked, a key press still sends an acknowledgement to the status register bit, although no action occurs. This means that the key can be mapped to perform a different function, or the usual function can be prevented.

Note: If an F-key is masked off, the action assigned to it will not appear at the bottom of the BTC06 screen of the Control Menu.

Registers 95-97: BTC06 Teach Pendant Status Registers

The bits in these registers are set when the corresponding keys are pressed on the teach pendant. They can be scanned in the user program or I-O mapper to detect system operations or to extend the teach pendant functionality to control the user program.

Note: These bits are all read-only.

95-1		96-1		97-1	
95-2		96-2		97-2	
95-3		96-3		97-3	
95-4		96-4		97-4	
95-5		96-5		97-5	
95-6		96-6		97-6	
95-7		96-7		97-7	
95-8		96-8		97-8	
95-9		96-9		97-9	Status of selected Jog System[1]
95-10		96-10		97-10	
95-11		96-11		97-11	Status of selected Jog Method[2]
95-12		96-12		97-12	
95-13		96-13		97-13	
95-14		96-14		97-14	
95-15		96-15		97-15	
95-16		96-16		97-16	[not used]

[1]The status of the selected jog system can be one of the following binary values in bits 97-9 and 97-10:

- 00 = world
- 01 = joint
- 10 = tool
- 11 = single axis

[2]The status of the selected jog method can be one of the following binary values in bits 97-11 and 97-12:

- 00 = incremental/large
- 01 = incremental/small
- 10 = continuous/fast
- 11 = continuous/slow

Registers 98, 99: Teach Pendant Control - Task A-B, C-D

The bits in the register can disable teach pendant control of the selected function for the corresponding tasks A - B, C-D

98-1	Block Task A Manual	99-1	Block Task C Manual
98-2	Block Task A Auto	99-2	Block Task C Auto
98-3	Block Task A Step	99-3	Block Task C Step
98-4	Block Task A Jog	99-4	Block Task C Jog
98-5	Block Task A Entry	99-5	Block Task C Entry
98-6	Block Task A Teach	99-6	Block Task C Teach
98-9	Block Task B Manual	99-9	Block Task D Manual
98-10	Block Task B Auto	99-10	Block Task D Auto
98-11	Block Task B Step	99-11	Block Task D Step
98-12	Block Task B Jog	99-12	Block Task D Jog
98-13	Block Task B Entry	99-13	Block Task D Entry
98-14	Block Task B Teach	99-14	Block Task D Teach

Register 400-405, 410-415: DEA (28/29/30) I/O Registers

DEA-28, DEA-29 & DEA-30 I/O cards are mapped into register space as shown in the following table. Cards may be located in a drive or in a CCD rack. Only one of each card may be used in a GPS system. When any of these cards are not present, its corresponding registers may be used by the application.

DEA 28/29/30 are special high density I/0 cards that plug into the CLC-D mezzanine bus. Each card has 32 inputs and 24 outputs that are read directly by the CLC. This offers an advantage over DEA 4/5/6 which require the SERCOS ring to be running before I/O can be read.

DEA Inputs

400	DEA-28	Inputs 1-16
401	DEA-28	Inputs 17-32
402	DEA-29	Inputs 1-16
403	DEA-29	Inputs 17-32
404	DEA-30	Inputs 1-16
405	DEA-30	Inputs 17-32

DEA Outputs

410	DEA-28	Outputs 1-16
411	DEA-28	Outputs 17-24 in bits 1-8
412	DEA-29	Outputs 1-16
413	DEA-29	Outputs 17-24 in bits 1-8
414	DEA-30	Outputs 1-16
415	DEA-30	Outputs 17-24 in bits 1-8

CLC Reserved Register Tables

Indramat reserves the first 99 registers for managing the CLC system, tasks, axes and DDS drives. User available I/O registers begin with register 100.

The following tables define the Indramat reserved registers and bits, and their associated default CLC label names.

Register	Register Label Name	Bit	Bit Label Name
01	System_Control	1	Parameter_Mode
		2	Bit_02
		3	nEmergency_Stop
		4	Bit_04
		5	Clear_All_Errors
		6	Pendant_Live_Man (switch)
		7	Bit_07
		8	Activate_Program
		9	Program_Select_LSB
		10	Program_Select_Bit_2
		11	Program_Select_Bit_3
		12	Program_Select_MSB
		13	Bit_13
		14	Pendant_Enable
		15	Pendant_Level_LSB
		16	Pendant_Level_MSB

Register	Register Label Name	Bit	Bit Label Name
02	TaskA_Control	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	Mode_Auto_nManual Bit_02 Bit_03 Single_Step Bit_05 Cycle_Start/Resume nTask_Stop Bit_08 Task_Event_Trigger Bit_10 Breakpoint_Enable Step_Seq_Step Step_Seq_Function Bit_14 Bit_15 Bit_16
03	TaskB_Control		(Same as Task A)
04	TaskC_Control		(Same as Task A)
05	TaskD_Control		(Same as Task A)
06	System_Diagnostic_Code	1-16	Bit_1-Bit_16
07	TaskA_Jog	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	Continuous_nStep Coord_Jog_Forward Coord_Jog_Reverse Jog_Type_LSB Jog_Type_MSB Distance_Speed Bit_07 Bit_08 Jog_X_Coord Jog_Y_Coord Jog_Z_Coord Jog_Joint_4 Jog_Joint_5 Jog_Joint_6 Bit_15 Bit_16
08	TaskB_Jog		(Same as Task A)
09	TaskC_Jog		(Same as Task A)
10	TaskD_Jog		(Same as Task A)

Register	Register Label Name	Bit	Bit Label Name
11	Axis1_Control	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	Disable_Axis Jog_Forward Jog_Reverse Synchronized_Jog Bit_05 Bit_06 Bit_07 Bit_08 Bit_09 Bit_10 Bit_11 Bit_12 Bit_13 Bit_14 Bit_15 Bit_16
12	Axis2_Control		(Same as Axis 1)
13	Axis3_Control		(Same as Axis 1)
14	Axis4_Control		(Same as Axis 1)
15	Axis5_Control		(Same as Axis 1)
16	Axis6_Control		(Same as Axis 1)
17	Axis7_Control		(Same as Axis 1)
18	Axis8_Control		(Same as Axis 1)

Register	Register Label Name	Bit	Bit Label Name
21	System_Status	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	Parameter_Mode Bit_02 Bit_03 Service_Chan_Ready Error Bit_06 Bit_07 Bit_08 Active_Program_LSB Active_Program_Bit_2 Active_Program_Bit_3 Active_Program_Bit_MSB TP_Password_Active Teach_Pendant Bit_15 Bit_16

Register	Register Label Name	Bit	Bit Label Name
22	TaskA_Status	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	Mode:Auto/nManual Bit_02 Bit_03 Single_Step Task_Error Task_Running Bit_07 Bit_08 Bit_09 Trace_Ready Breakpoint_Reached Bit_12 Bit_13 Bit_14 Bit_15 Bit_16
23	TaskB_Status		(Same as Task A)
24	TaskC_Status		(Same as Task A)
25	TaskD_Status		(Same as Task A)

Register	Register Label Name	Bit	Bit Label Name
27	Eagle Module Inputs	1-8	Input1-8
		9-12	Future Sp_Function
28	Eagle Module Outputs	1-4	Output1-4
29	ELS Master Control	4	Real Master: Primary/!Secondary
		5	Master Type: Virtual/nReal
30	ELS Master Status	4	Real Master: Primary/nSecondary
		5	Master Type: Virtual/nReal
		6	Virtual Master Enabled
		7	Virtaul Master at Programmed Speed
		9	Master Direction: Positive/nNegative
		10	Master Stopped
		11	Position Difference Monitoring
		12	Position Difference Monitoring
31	Axis1_Status	1	Bit_01
		2	Jogging_Forward
		3	Jogging_Reverse
		4	Phase_Adjusted
		5	ELS_Enabled
		6	ELS_Secondary_Mode
		7	Axis_in_Position
		8	Axis_Aligned
		9	Bit_09
		10	Bit_10
		11	Axis_Halted
		12	Class_3_Status
		13	Class_2_Warning
		14	Shutdown_Error
		15	Drive_Ready_LSB
		16	Drive_Ready_MSB
32	Axis2_Status		(Same as Axis 1)
33	Axis3_Status		(Same as Axis 1)
34	Axis4_Status		(Same as Axis 1)
35	Axis5_Status		(Same as Axis 1)
36	Axis6_Status		(Same as Axis 1)
37	Axis7_Status		(Same as Axis 1)
38	Axis8_Status		(Same as Axis 1)

Register	Register Label Name	Bit	Bit Label Name
40	DDS1_1IN	1 . . . 16	Bit_01 . . . Bit_16
41	DDS1-1OUT		(same)
42	DDS1_2IN		(same)
43	DDS1_2OUT		(same)
44	DDS1_3IN		(same)
45	DDS1_3OUT		(same)

Registers 40 through 45 access inputs and outputs provided by Indramat DDS I/O cards installed in the DDS drive 1 enclosure.

Register	Register Label Name	Bit	Bit Label Name
46	DDS2_1IN	1 . . . 16	Bit_01 . . . Bit_16
47	DDS2-1OUT		(same)
48	DDS2_2IN		(same)
49	DDS2_2OUT		(same)
50	DDS2_3IN		(same)
51	DDS2_3OUT		(same)

Registers 46 through 51 access inputs and outputs provided by Indramat DDS I/O cards installed in the DDS drive 2 enclosure.

Register	Register Label Name	Bit	Bit Label Name
52	DDS3_1IN	1 . . . 16	Bit_01 . . . Bit_16
53	DDS3-1OUT		(same)
54	DDS3_2IN		(same)
55	DDS3_2OUT		(same)
56	DDS3_3IN		(same)
57	DDS3_3OUT		(same)

Registers 52 through 57 access inputs and outputs provided by Indramat DDS I/O cards installed in the DDS drive 3 enclosure.

Register	Register Label Name	Bit	Bit Label Name
58	DDS 4_1IN	1 . . . 16	Bit_01 . . . Bit_16
59	DDS 4-1OUT		(same)
60	DDS 4_2IN		(same)
61	DDS 4_2OUT		(same)
62	DDS 4_ 3IN		(same)
63	DDS 4_ 3OUT		(same)

Registers 58 through 63 access inputs and outputs provided by Indramat DDS I/O cards installed in the DDS drive 4 enclosure.

Register	Register Label Name	Bit	Bit Label Name
64	DDS 5_1IN	1 . . . 16	Bit_01 . . . Bit_16
65	DDS 5-1OUT		(same)
66	DDS 5_2IN		(same)
67	DDS 5_2OUT		(same)
68	DDS 5_ 3IN		(same)
69	DDS 5_ 3OUT		(same)

Registers 64 through 69 access inputs and outputs provided by Indramat DDS I/O cards installed in the DDS drive 5 enclosure.

Register	Register Label Name	Bit	Bit Label Name
70	DDS 6_1IN	1 . . . 16	Bit_01 . . . Bit_16
71	DDS 6-1OUT		(same)
72	DDS 6_2IN		(same)
73	DDS 6_2OUT		(same)
74	DDS 6_ 3IN		(same)
75	DDS 6_ 3OUT		(same)

Registers 70 through 75 access inputs and outputs provided by Indramat DDS I/O cards installed in the DDS drive 6 enclosure.

Register	Register Label Name	Bit	Bit Label Name
76	DDS 7_1IN	1 . . . 16	Bit_01 . . . Bit_16
77	DDS 7-1OUT		(same)
78	DDS 7_2IN		(same)
79	DDS 7_2OUT		(same)
80	DDS 7_ 3IN		(same)
81	DDS 7_ 3OUT		(same)

Registers 76 through 81 access inputs and outputs provided by Indramat DDS I/O cards installed in the DDS drive 7 enclosure.

Register	Register Label Name	Bit	Bit Label Name
82	DDS 8_1IN	1 . . . 16	Bit_01 . . . Bit_16
83	DDS 8-1OUT		(same)
84	DDS 8_2IN		(same)
85	DDS 8_2OUT		(same)
86	DDS 8_ 3IN		(same)
87	DDS 8_ 3OUT		(same)
88	Task A Extend Event Control		
89	Task A Extend Event Control		

Registers 82 through 89 access inputs and outputs provided by Indramat DDS I/O cards installed in the DDS drive 8 enclosure.

Register	Register Label Name	Bit	Bit Label Name
90	Latch	1 . . . 16	Bit_01 . . . Bit_16
91	Unlatch		(same)
95	Teach Pendant Status	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	F1 Key F2 Key F3 Key F4 Key F5 Key F6 Key F7 Key F8 Key Help Key 7 Key 8 Key 9 Key Teach Key Task Key Axis Plus Key Axis Minus Key
96	Teach Pendant Status	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	Esc Key 4 Key 5 Key 6 Key A Plus Key A Minus Key X Plus Key X Minus Key Edit Key 1 Key 2 Key 3 Key B Plus Key B Minus Key Y Plus Key Y Minus Key

Register	Register Label Name	Bit	Bit Label Name
97	Teach Pendant Status	1	Enter Key
		2	Decimal Key
		3	Zero Key
		4	Plus Minus Key
		5	C Plus Key
		6	C Minus Key
		7	Z Plus Key
		8	Z Minus Key
98	Teach Pendant Control	1	Block Task A Manual
		2	Block Task A Auto
		3	Block Task A Step
		4	Block Task A Jog
		5	Block Task A Entry
		6	Block Task A Teach
		7	Bit 7
		8	Bit 8
		9	Block Task B Manual
		10	Block Task B Auto
		11	Block Task B Step
		12	Block Task B Jog
		13	Block Task B Entry
		14	Block Task B Teach
		15	Bit 15
		16	Bit 16
99	Teach Pendant Control	1	Block Task C Manual
		2	Block Task C Auto
		3	Block Task C Step
		4	Block Task C Jog
		5	Block Task C Entry
		6	Block Task C Teach
		7	Bit 7
		8	Bit 8
		9	Block Task D Manual
		10	Block Task D Auto
		11	Block Task D Step
		12	Block Task D Jog
		13	Block Task D Entry
		14	Block Task D Teach
		15	Bit 15
		16	Bit 16

Register	Register Label Name	Bit	Bit Label Name
100 101	User Inputs Reg 1 User Inputs Reg 2 These registers are reserved if an Indramat supported I/O subsystem and physical I/O drivers are used. *For specific assignments see the appropriate I/O system section.* If Indramat I/O drivers are not used, the registers are available to the user.		

Register	Register Label Name	Bit	Bit Label Name
120 121	User Outputs Reg 1 User Outputs Reg 2 These registers are reserved if an Indramat supported I/O subsystem and physical I/O drivers are used. *For specific assignments see the appropriate I/O system section.* If Indramat I/O drivers are not used, the registers are available to the user.		

3 Parameters

3.1 Overview

The CLC card uses user-configurable parameters to adapt to a specific application. Basic communication and initialization parameters for the CLC and DDS digital drives must be entered before it is possible to operate or program a CLC system.

System-builders must enter certain parameters describing the mechanical characteristics of their unique application. Parameters may be specific to an application and/or a single machine installation. These parameters specify the machine's limitations such as maximum velocity and acceleration. Other parameters are used to specify the mechanical characteristics of the system, such as the ratio between motor revolutions and shaft rotation or slide travel.

The CLC system divides parameters into four functionally related classes: System (Card), Task, Axis and Drive. The System parameters contain setup and status parameters for the CLC card. The Task and Axis parameters are associated with user programs. The System, Task and Axis parameters are downloaded and stored in the CLC card. Digital Drive parameters are stored in each SERCOS-compatible digital drive and are needed to configure a motion system.

Each parameter within a Class also has a set and number identifier. All parameter elements (data, name, units) can be accessed using the Visual Motion Windows application or from a user specified interface such as an ASCII terminal (***see Appendix B. Direct ASCII Communication***). Parameters in this chapter are described using the following format:

Parameter Identification

All parameters on the CLC have an identification number (IDN): X-s-nnnn

Where:

X= parameter type:

- C= CLC System Parameter
- T= CLC Task Parameter
- A= CLC Axis Parameter
- S= Drive SERCOS Standard Parameter
- P= Drive Product-specific Parameter

s= parameter set, if drive has different sets, a number between 0 - 7

nnnn= parameter number from 1 - 4095.

Examples:

C-0-0001, is CLC system parameter 1

P-1-0001, is a product-specific drive 1, parameter 1.

Class 'C' includes the system setup and status parameters for the CLC card.

Certain status parameters are stored on the CLC card in non-volatile memory, while other status parameters are only temporary indicators and are lost when the system is powered off. Class 'C' parameters have only one set and are indicated by the number '0'.

(Examples: *C-0-0nnn*, etc.)

Class 'T' parameters include setup and status information for each task. One of the four task parameter sets is indicated by an ASCII character 'A' through 'D' corresponding to one of the user program tasks: A, B, C or D.

(Example: Task A *01 T-0-nnnn*, Task C *03 T-0-nnnn*, etc.)

Class 'A' includes the parameters used to configure and display information about the CLC Axes. A maximum of 40 different sets of parameters are possible. A set corresponding to one of the eight axes is identified by a single digit in the range 1-40.

(Examples: Axis *01 A-0-nnnn*, Axis *03 A-0-nnnn*, etc.)

Class 'D' parameters, the DDS drive parameters, are accessed through the SERCOS Service Channel. The drive parameter sets 1 through 8 directly correspond to the SERCOS drive address 1 through 8 and the CLC Axes 1-8.

(Examples: Drive *01 S-0-nnnn*, Drive *03 P-0-nnnn*)

Most servo and position loop parameters are contained on the SERCOS-compatible digital drive. These parameters include: Feed Constant, Kv Factor, In-Position Window, Monitor Window, and all homing parameters. Acceleration, Deceleration, and Jerk are defined in the user program and limits for these are set in the drive. A list of most often changed drive parameters is included at the end of this section. For a complete description of the parameters, refer to the Drive Manual.

Some parameters may be accessed by the user program through the Visual Motion parameter transfer commands or the text programming language Parameter commands. Visual Motion also allows access to specific groups of parameters for editing and archiving. See the relevant chapters on Icon Programming or Text Language Programming, and the section on Direct ASCII Communication for information on transferring parameters.

Lists of all parameters and all non-volatile parameters are available through the CLC's serial communication port. Transferring parameters using direct communication follows the same format, with various subclasses for names, units, limits, and data. See the chapter on Direct ASCII Communication for more information on the protocol and syntax for direct communication.

3.2 Parameter Transfer Commands

A parameter transfer capability is built into the CLC's programming language. Parameter transfers allow the monitoring of system values and SERCOS command execution. The System, Axis, and Task parameter descriptions in this section specify the type of data (Float, Integer, etc.) set or returned by the PARAMETER/BIT, PARAMETER/GET, PARAMETER/INIT, PARAMETER/SET text language commands.

See Chapter 7 - Text Language Programming, Chapter 6 - Icon Programming and the Parameter Transfer Icon.

All CLC and drive parameters of the floating point type may be read into a CLC floating point variable. All other parameters, unless they are a String or List type, can be read into a CLC integer variable. The CLC issues a runtime error if values are transferred to/from the wrong type of variable. The parameters classified as "Read/write at any time" may be changed by a user program.

To transfer SERCOS parameters not listed in this section, refer to the DDS and SERCOS manuals for a specific parameter's type of data. In general: feedback, tuning, and measurement values are floating point; procedure commands are integer. Use the List Parameter capability to obtain a list containing specific parameters, then find the types of CLC variables that can contain specific parameters.

The full capability of the SERCOS drive is accessible through a user program. Procedural commands such as homing can be performed from a user program by initiating the SERCOS homing sequence, then testing the appropriate parameter to determine when the operation has completed. The SERCOS probing functions may be used to implement registration to a mark, or for accurate position measurements. In addition, some of the limits and offsets can be set up at program initialization, or a user program may be written to dynamically tune a drive based on feedback of the system's performance.

3.3 List of Parameters

This chapter includes specific information on the CLC System, Drive, Axis, and Task parameters listed below. Additional information on SERCOS and DDS parameters may be found in the respective SERCOS and DDS manuals.

System (Card) Parameters - Class C, Set 0

CLC system parameters include System Setup, Jogging and Display, Program Management, System Status, Electronic Line Shaft, VME, Pendant, and Parameter Lists.

System Setup Parameters

C-0-0001 Language Selection
C-0-0002 Unit Number
C-0-0003 Serial Port A Setup
C-0-0004 Serial Port B Setup
C-0-0005 I/O Device Selection
C-0-0006 I/O Device Setup
C-0-0007 I/O Device Direction
C-0-0008 I/O Device Base Address
C-0-0009 Error Reaction Mode
C-0-0010 System Options

C-0-0012 Serial Port B Device Type
C-0-0013 Serial Port A Mode
C-0-0014 Serial Port B Mode
C-0-0016 Communication Time-out Period
C-0-0020 Transmitter Fiber Optic Length
C-0-0021 User Watchdog Timer
C-0-0022 User Watchdog Task ID
C-0-0030 Option Cards Active
C-0-0031 Option Card Status

Jogging and Display Parameters

C-0-0042 World Large Increment
C-0-0043 World Small Increment
C-0-0045 World Fast Jog Speed
C-0-0046 World Slow Jog Speed
C-0-0052 Axis Large Increment
C-0-0053 Axis Small Increment
C-0-0055 Axis Fast Jog Velocity
C-0-0056 Axis Slow Jog Velocity

Program Management Parameters

C-0-0090 Download Block Size
C-0-0091 Total Program Memory
C-0-0092 Available Program Memory
C-0-0093 Contiguous Program Memory
C-0-0094 Maximum Executable Program Size
C-0-0095 Path Planner to SERCOS Time Factor
C-0-0098 Initialization Delay
C-0-0099 Minimum SERCOS cycle Time

System Status Parameters

C-0-0100 CLC Firmware Version
C-0-0101 CLC Hardware Version
C-0-0102 CLC Version Date
C-0-0103 Maximum Number of Axes Allowed
C-0-0111 Data Backup Selection
C-0-0112 Data Backup Command
C-0-0120 Operating Mode
C-0-0121 SERCOS Communication Phase
C-0-0122 Diagnostic Message
C-0-0123 Diagnostic Code
C-0-0124 Extended Diagnostic
C-0-0125 System Timer Value

C-0-0126 Date and Time

Electronic Line Shaft Parameters *(See also C-0-1000 to C-0-1568)*

C-0-0150 Master 1 Type
C-0-0151 Master 1 Drive Address
C-0-0152 ELS Master Task Assignment
C-0-0153 Virtual Master Programmed Velocity
C-0-0154 Virtual Master Programmed Acceleration
C-0-0155 Virtual Master Programmed Deceleration
C-0-0156 Virtual Master E-Stop Deceleration
C-0-0157 ELS Master Current Position
C-0-0158 ELS Master Current Velocity
C-0-0159 Master 1 Zero Velocity Window
C-0-0160 Virtual Master Maximum Jog Velocity
C-0-0161 Master 1 Ratio Input
C-0-0162 Master 1 Ratio Output
C-0-0164 Master 1 Encoder Type
C-0-0170 PLS 1 Mask Register

VME Parameters

C-0-0200 Bus Arbitration Mode
C-0-0201 Bus Release Mode
C-0-0202 VME / CLC Interface
C-0-0203 VME Arbiter Status
C-0-0210 Unit 0 Short Address Page
C-0-0211 Unit 0 Access
C-0-0212 Unit 0 Window Address Page
C-0-0215 Unit 1 Short Address Page
C-0-0216 Unit 1 Access
C-0-0217 Unit 1 Window Address Page
C-0-0220 Unit 2 Short Address Page
C-0-0221 Unit 2 Access
C-0-0222 Unit 2 Window Address Page
C-0-0225 Unit 3 Short Address Page
C-0-0226 Unit 3 Access
C-0-0227 Unit 3 Window Address Page
C-0-0230 Unit 4 Short Address Page
C-0-0231 Unit 4 Access
C-0-0232 Unit4 Window Address Page
C-0-0235 Unit 5 Short Address Page
C-0-0236 Unit 5 Access
C-0-0237 Unit 5 Window Address Page
C-0-0240 Unit 6 Short Address Page

C-0-0241 Unit 6 Access
C-0-0242 Unit 6 Window Address Page
C-0-0245 Unit 7 Short Address Page
C-0-0246 Unit 7 Access
C-0-0247 Unit 7 Window Address Page
C-0-0250 Unit 8 Short Address Page
C-0-0251 Unit 8 Access
C-0-0252 Unit 8 Window Address Page
C-0-0255 Unit 9 Short Address Page
C-0-0256 Unit 9 Access
C-0-0257 Unit 9 Window Address Page
C-0-0260 Unit 10 Short Address Page
C-0-0261 Unit 10 Access
C-0-0262 Unit 10 Window Address Page
C-0-0265 Unit 11 Short Address Page
C-0-0266 Unit 11 Access
C-0-0267 Unit 11 Window Address Page
C-0-0270 Unit 12 Short Address Page
C-0-0271 Unit 12 Access
C-0-0272 Unit 12 Window Address Page
C-0-0275 Unit 13 Short Address Page
C-0-0276 Unit 13 Access
C-0-0277 Unit 13 Window Address Page
C-0-0280 Unit 14 Short Address Page
C-0-0281 Unit 14 Access
C-0-0282 Unit 14 Window Address Page
C-0-0285 Unit 15 Short Address Page
C-0-0286 Unit 15 Access
C-0-0287 Unit 15 Window Address Page

Pendant

C-0-0801 Pendant Protection Level 1 Password
C-0-0802 Pendant Protection Level 2 Password
C-0-0803 Pendant User Accessible Floats Section
C-0-0804 Pendant User Accessible Integers Section
C-0-0805 Pendant Start of User Accessible Registers
C-0-0806 Pendant End of User Accessible Registers
C-0-0807 Pendant Password Timeout
C-0-0810 BTC06 message and prompt control word
C-0-0811 User Task Controlled Menu ID for BTC06
C-0-0812 User Task Controlled Task ID for BTC06
C-0-0813 User Task Controlled Axis Number for BTC06
C-0-0814 BTC06 Data Transaction Word

C-0-0090 Exit to Monitor Prompt
C-0-0991 System Error Information

Electronic Line Shaft Parameters *(See also C-0-0150 to C-0-0164)*

C-0-1000 ELS Primary Master
C-0-1001 ELS Secondary Master
C-0-1010 Master Switching Threshold
C-0-1011 Diff. Between Real Master Positions
C-0-1012 Master Synchronization Acceleration
C-0-1013 Master Synchronization Time Constant
C-0-1014 Master Synchronization Velocity Window
C-0-1015 Virtual Master Current Position
C-0-1016 Virtual Master Current Velocity
C-0-1017 Position Difference Selection
C-0-1018 Position Monitoring Window
C-0-1019 Position Difference
C-0-1508 Master 1 Filter Cutoff Frequency
C-0-1509 Master 1 Filter Type
C-0-1517 Master 1 Current Position
C-0-1518 Master 1 Current Velocity
C-0-1550 Master 2 Type
C-0-1551 Master 2 Drive Address
C-0-1552 Master 2 Encoder Type
C-0-1554 Master 2 Ratio Input
C-0-1555 Master 2 Ratio Output
C-0-1556 Master 2 Zero Velocity Window
C-0-1558 Master 2 Filter Cutoff Frequency
C-0-1559 Master 2 Filter Type
C-0-1567 Master 2 Current Position
C-0-1568 Master 2 Current Velocity

System Parameter Lists

C-0-2000 List of All Parameters
C-0-2001 List of Required Parameters
C-0-2010 List of SERCOS Devices
C-0-2011 List of SERCOS Drives
C-0-2012 List of SERCOS I-O Stations
C-0-2013 SERCOS I-O Configuration List
C-0-2014 Start of SERCOS I-O Station Registers
C-0-2015 Registers Allocated per I-O Station
C-0-2020 Diagnostic Log List
C-0-2021 Diagnostic Log Options
C-0-2501 Oscilloscope signal 1 type

C-0-2502 Oscilloscope signal 2 type
C-0-2503 Oscilloscope signal 3 type
C-0-2504 Oscilloscope signal 1 id number
C-0-2505 Oscilloscope signal 2 id number
C-0-2506 Oscilloscope signal 3 id number
C-0-2507 Oscilloscope signal 1 axis number
C-0-2508 Oscilloscope signal 2 axis number
C-0-2509 Oscilloscope signal 3 axis number
C-0-2510 Oscilloscope sample rate
C-0-2511 Oscilloscope signal 1 list
C-0-2512 Oscilloscope signal 2 list
C-0-2513 Oscilloscope signal 3 list
C-0-2514 Oscilloscope sample count
C-0-2515 Oscilloscope trigger post-count
C-0-2516 Oscilloscope trigger type
C-0-2517 Oscilloscope trigger id number
C-0-2518 Oscilloscope trigger axis or mask
C-0-2519 Oscilloscope trigger level or mask
C-0-2520 Oscilloscope trigger mode
C-0-2521 Oscilloscope trigger source
C-0-2522 Oscilloscope trigger control word
C-0-2523 Oscilloscope trigger status word
C-0-2600 Field Bus Object Mapper
C-0-2610 Interbus-S Card Enable
C-0-2630 Profibus Slave Device Address
C-0-2631 Length of Profibus Parameter Channel
C-0-2632 Profibus Multiplex Control Word
C-0-2634 Profibus PD Channel Length
C-0-2640 Profibus PD Objects
to
C-0-2698 Profibus PD Objects
C-0-2700 Field Bus COM Object Mapper
C-0-3000 I-O Mapper Program
C-0-3001 I-O Mapper Options
C-0-3003 I-O Mapper Total Operations
C-0-3004 I-O Mapper File Size
C-0-3005 I-O Mapper Executable Size
C-0-3100 Cam Tags
C-0-3101 Cam Table 1
C-0-3102 Cam Table 2
C-0-3103 Cam Table 3
C-0-3104 Cam Table 4
C-0-3105 Cam Table 5

C-0-3106 Cam Table 6
C-0-3107 Cam Table 7
C-0-3108 Cam Table 8

Task Parameters- -Class T

CLC user task parameters include Task Setup, Task Status, Coordinated Motion and Robotics, Coordinated Status, and Parameter Lists.

Task Setup Parameters

T-0-0001 Task Motion Type
T-0-0002 Task Options

Coordinated Motion

T-0-0005 World Position Units
T-0-0010 Kinematic Number
T-0-0011 Coordinated X-axis
T-0-0012 Coordinated Y-axis
T-0-0013 Coordinated Z-axis
T-0-0020 Maximum Path Speed
T-0-0021 Maximum Acceleration
T-0-0022 Maximum Deceleration
T-0-0023 Look Ahead Distance
T-0-0024 Velocity Override
T-0-0025 Maximum Jog Increment
T-0-0026 Maximum Jog Velocity

Robotics

T-0-0035 Relative Point Used for Origin
T-0-0036 Relative Point Used for Tool Frame
T-0-0050 Kinematic Value 1
T-0-0051 Kinematic Value 2
T-0-0052 Kinematic Value 3
T-0-0053 Kinematic Value 4
T-0-0054 Kinematic Value 5
T-0-0055 Kinematic Value 6
T-0-0056 Kinematic Value 7
T-0-0057 Kinematic Value 8
T-0-0058 Kinematic Value 9
T-0-0059 Kinematic Value 10

Coordinated Motion Status

T-0-0100 Target Point Number
T-0-0101 Segment Status
T-0-0102 Rate Limit Status
T-0-0111 Current X Position
T-0-0112 Current Y Position
T-0-0113 Current Z Position

Task Status

T-0-0120 Task Operating Mode
T-0-0122 Task Diagnostic Message
T-0-0123 Task Status Message
T-0-0130 Current Instruction Pointer
T-0-0131 Current Instruction
T-0-0132 Instruction Pointer at Error
T-0-0133 Composite Instruction Pointer
T-0-0135 Current Subroutine
T-0-0136 Stack Variable Data
T-0-0137 Subroutine Breakpoint
T-0-0138 Sequencer Information
T-0-0200 Last Active Event Number

Task Parameter Lists

T-0-2000 List of All Parameters
T-0-2001 List of Required Parameters

Axis Parameters - Class A

CLC axis parameters include Setup Parameters, Status Parameters, Electronic Line Shaft, Feedback Capture, Optional SERCOS Data, and Parameter lists.

Axis Setup Parameters

A-0-0001 Task Assignment
A-0-0002 Type of Positioning
A-0-0003 Axis Motion Type
A-0-0004 Axis Options
A-0-0005 Linear Position Units
A-0-0006 Reference Options
A-0-0007 Configuration Mode
A-0-0009 Drive PLS Register
A-0-0010 Start of Drive I/O
A-0-0011 Drive I-O Card 1 Setup
A-0-0014 Drive I-O Card 2 Setup
A-0-0017 Drive I-O Card 3 Setup
A-0-0020 Maximum Velocity
A-0-0021 Maximum Acceleration
A-0-0022 Maximum Deceleration
A-0-0023 Jog Acceleration
A-0-0025 Maximum Jog Increment
A-0-0026 Maximum Jog Velocity
A-0-0030 Ratio Mode Master Axis
A-0-0031 CLC Cam/Ratio Master Factor (N)
A-0-0032 CLC Cam/Ratio Slave Factor (M)
A-0-0033 CLC Cam Stretch Factor (H)
A-0-0034 CLC Cam Currently Active
A-0-0035 CLC Cam Position Constant (L)
A-0-0036 Ratio Mode Encoder Type
A-0-0037 Ratio Mode Step Rate
A-0-0038 Ratio Mode Options

Axis Status Parameters

A-0-0100 Target Position
A-0-0101 Commanded Position
A-0-0102 Feedback Position
A-0-0110 Programmed Velocity
A-0-0111 Commanded Velocity
A-0-0112 Feedback Velocity
A-0-0120 Programmed Acceleration
A-0-0131 SERCOS Control Word

A-0-0132 SERCOS Status Word
A-0-0133 AT Error Count
A-0-0140 Mfg. Class 3 Status Word
A-0-0141 Torque Mode Commanded Torque
A-0-0142 Torque Feedback (cyclic)
A-0-0145 Current Motion Type

Axis Parameters - Electronic Line Shaft

A-0-0150 Programmed Ratio Adjust
A-0-0151 Programmed Phase Offset
A-0-0153 CLC Phase Adjust Average Velocity
A-0-0155 CLC Phase Adjust Time Constant
A-0-0157 Current Phase/ CLC Cam Master Offset
A-0-0159 Ratio Adjust Step Rate
A-0-0160 Commanded Ratio Adjust
A-0-0161 CLC Cam Programmed Slave Adjust
A-0-0162 CLC Cam Current Slave Adjust (Sph)
A-0-0163 CLC Cam Output Position
A-0-0164 ELS Options

Axis Feedback Capture (Registration)

A-0-0170 Probe Configuration Status
A-0-0171 Probe 1 Positive Captured Value
A-0-0172 Probe 1 Negative Captured Value
A-0-0173 Probe 2 Positive Captured Value
A-0-0174 Probe 2 Negative Captured Value

Optional SERCOS Data

A-0-0180 Optional Command ID #1
A-0-0181 Optional Command ID #2
A-0-0182 Optional Command ID #3
A-0-0185 Optional Feedback ID #1
A-0-0186 Optional Feedback ID #2
A-0-0190 Command Data #1
A-0-0191 Command Data #2
A-0-0192 Command Data #3
A-0-0195 Feedback Data #1
A-0-0196 Feedback Data #2

Axis Parameter Lists

A-0-2000 List of All Parameters
A-0-2001 List of Required Parameters

3.4 System Parameters

Class C, Set 1

The system parameter set includes setup parameters for system configuration, program options, serial interface, and I/O options. Also included are status parameters such as operating mode, and diagnostic messages.

System Setup (0001-0031)

C-0-0001 Language Selection

Selections:	0 = German
	1 = English
	2 = French
	3 = Spanish (Future)
Default:	1
Attributes:	Integer, Read/write at any time

Parameter C-0-0001 selects the language in which CLC diagnostic and parameter messages are sent. The Language parameter for each Indramat DDS drive is also set automatically.

C-0-0002 Unit Number

Selections:	0-31
Attributes:	VME: Integer, Read-only (set by switch on board)
Other:	Integer, Read/Write

The CLC permits cards to communicate over the VME backplane. This parameter will set the network address of each card so that one Host can communicate with up to 32 cards. When changing the unit number in this parameter be sure to also change the card number in the card selection setup menu. The CLC card will then need to be reset by power cycling.

Note: *This applies to the CLC-V only. For CLC-D, it is software selectable since multiple cards can be addressed with an RS-485 connection.*

C-0-0003 Serial Port A Setup

Format:	Baud Rate + Option
Valid Baud Rates:	300, 1200, 2400, 4800, 9600, 19200, 38400
Valid Options:	1 = checksum on

Default: 9601 (9600 baud, checksum on)

Attributes: Integer, Read/write at any time

This parameter sets the baud rate and options for Serial Port A (X27 on CLC/D and CLC/P), which communicates with a PC, a terminal, or any device that follows the CLC ASCII Host Protocol. The port always operates with 8 data bits, 1 stop bit, and no parity.

If the checksum option is enabled, the CLC will send a checksum as described in the Host Protocol Description, and will check the checksum in data received from the host.

The options are added together with the baud rate. For example, to set 9600 baud, checksum on, the parameter would be set to (9600 + 1)= 9601.

C-0-0004 Serial Port B Setup

Format:	Baud Rate + Options
Valid Baud Rates:	1200, 2400, 4800, 9600
Options for Host Protocol:	1 = checksum on
	2 = send CLC status
Default:	9601 (9600 + checksum on)
Attributes:	Integer, Read/write at any time

This parameter sets the baud rate and options for Serial Port B (X28 on CLC/D and CLC/P). The device connected to this port is selected in parameter C-0-0012 Serial Port B Device Type. The port always operates with 8 data bits, 1 stop bit, and no parity. See the description of parameter C-0-0003 Serial Port A Setup for Host Protocol options.

Note: *The baud rate of this port is limited to 9600. For high-speed host communications, port A should be used.*

C-0-0005 I/O Device Selection

Selections:	0= no parallel I/O
	1= Xycom XVME-202 PAMUX
	2= Pentland MPV922 VME
	3= Xycom XVME-201 Digital
	4 = Indramat Demo
	5 = Xycom XVME-244 Digital
Default:	0
Attributes:	Integer, Read/write at any time

The CLC can communicate with one parallel I/O device in addition to I/O on the DDS digital drives. Each device is mapped into a set of I/O registers. ***See Chapter 2. CLC I/O Systems for more information***.

The Indramat Demo selects a XVME-201 I/O card and inverts the signal

states. This selection is not recommended for field use because the I/O lines are set to a high state on reset.

C-0-0006 I/O Device Setup

Selections: LS bit selects lowest address
Default: 0000000000000000
Attributes: Binary Word, Read/write at any time

Sixteen binary bits select the address or other setup information for the device. The format of this parameter depends on the device selected.

For the Xycom PAMUX, the 16 bits correspond to stations 0x0000-0x000F.

For Xycom Digital, they correspond to the two I/O connectors on up to 8 cards.

See Chapter 2. CLC I/O Systems for more information..

C-0-0007 I/O Device Direction

Selections: LS bit selects direction for lowest address
Default: 0000000000000000
Attributes: Binary Word, Read/write at any time

The amount of I/O can be configured for each device. For Xycom I/O devices, a "0" selects a 24 bit station to have 16 inputs and 8 outputs, a "1" selects 16 outputs and 8 inputs.

See Chapter 2. CLC I/O Systems for more information.

C-0-0008 I/O Device Base Address

Selections: VME Short Address for selected I/O device
Default: 0x0000
Attributes: Hex Word, Read/write at any time

This parameter selects the VME bus base address for the specified parallel I/O card. VME parallel I/O is accessed using the 16-bit short address space. This parameter sets the starting address for an I/O interface and must correspond to the address settings of the interface card. Refer to the appropriate CLC I/O Systems and VME memory sections for more information.

C-0-0009 Error Reaction Mode

Codes: 0 = disable drive when it is finished with its reaction selected in P-0-0119. ***See Drive Help for more information.***
1 = decelerate to zero velocity, then disable drives
Default: 0

Attributes: Integer, write only in parameter mode

This parameter selects the CLC's reaction to a fatal shutdown error on the card or the drive. Fatal errors include Emergency Stop, Zone Violation, or Drive Class 1 Shutdowns.

When this parameter is set to 0 (default), the drives are immediately disabled. Velocity is immediately set to zero and the brakes are engaged.

When this parameter is set to 1, all motion will come to a controlled stop before the drives are disabled. In the case of coordinated and single-axis motion, the maximum task or axis deceleration will be used. For an ELS virtual master, the E-Stop deceleration will be used.

This Parameter works with drive error reaction mode parameters: P-0-0117, P-0-0118, P-0-0119.

C-0-0010 System Options

Units:	none
Default:	0000000000000000
Attributes:	Binary, Write in parameter mode only
Note:	Bit 1 is the rightmost bit

This parameter sets several options for the CLC system and for the SERCOS ring.

Bit 1: Simulation Mode

0= normal drive operation

1= simulation mode, do not scan for drives

When this bit is set to 1, the axes will be simulated. The SERCOS ring will not be scanned for drives, and the drive enable bits will be ignored. This mode is useful for simulating coordinated motion when the CLC is not connected to the actual system, or when a program does not contain any axes. All axis and task status parameters are simulated. Drive parameters and I-O since they require a SERCOS drive, are not simulated. Any drive-controlled motion (homing, single-axis, etc.) is also not simulated.

Bits 2-3: Phase 2 SERCOS I-O Update

bit3 bit2

0 0 = 500ms update

0 1 = 10 second update

1 x = no update

In SERCOS Phase 2 (Parameter Mode), all drive-resident I-O cards are scanned every 500ms by default (bit=0). This can slow down communications when downloading parameter lists from the drives, depending upon how many I-O cards there are. If it is not critical to have

fast updates of I-O in Phase 2, set these bits to a nonzero value. The I-O will be scanned every 10 seconds or not at all, which speeds up the communication to the user interface.

Bit 4: SERCOS synchronization clock

On CLC-V 2.3 hardware only

0= SERCOS clock is independent

1= Use clock generated by Card 0

When this bit is set to (0), each CLC card has an independent SERCOS ring. The clocks between these rings are not synchronized.

When this bit is set to (1), synchronization of multiple SERCOS rings across the VME backplane is accomplished by using a common SERCOS clock driven by one card. The card with Address 0 always drives the clock. This option should be enabled for applications that require synchronization between all axes in the system. All cards that use the common clock must have this bit set to (1).

Notes:

1. If card 0 does not exist or has a faulty interface board, the SERCOS rings for all other cards on the backplane will stay in phase 0 and the CLC will issue a SERCOS Controller Error.
2. All cards on the bus that use a common synchronization clock must have the same SERCOS cycle time set in Parameter C-0-0099 Minimum SERCOS Cycle Time. Automatic configuration of cycle time is not performed by the CLC, but a diagnostic indicates the cycle time that should be set in this parameter.
3. The LED that blinks on the CLC front panel is not synchronized with the SERCOS ring and is used for diagnostic purposes only. The LED's on the CLC's will not blink in unison.

Bits 5: Enable 4MBps SERCOS Transmission

When bit 5 is set to 1, the CLC transmits at 4MBps; when it is 0, 2MBps. All Indramat drives except DIAX01 and DIAX02 can be set for 4MBps, via a switch on the SERCOS interface card. The only applications that may require 4MBps are those with more than 8 drives. In all other cases, the 2MBps rate should be used. The transmission rate has no effect on CLC or drive performance unless a smaller SERCOS cycle time is needed. The minimum SERCOS cycle time is 2ms.

Bits 6-9: reserved for future use

Bit 10: System Timing Monitoring

System timing monitoring can be disabled by setting C-0-0010 bit 10 to (1). This allows users to continue running the system until he/she gets assistance for solving the problem.

Bit 11: Jog in Auto Mode

When system parameter C-0-0010, bit 11 is set to (1), jogging can be performed in auto. mode or when a task is running. This allows continuous or incremental axis motion to be started and stopped by simply setting its jog bits and parameters.

Any axis that is currently in single-axis or velocity mode can be jogged with the axis jog bits. The ELS virtual master can be jogged with the Task Jog Register coordinated Jog Forward and Jog Reverse bits.

Bit 12: Prioritized Service Channel

0 = Don't prioritize the service channel (default)

1 = User tasks have priority over lists from user interface

Only one task or user interface can use a drive's SERCOS service channel at a time. When drive parameter lists, long text strings, or oscilloscope data are transferred from one task, the other task could be suspended from 100ms to 5 seconds.

If the timing for user task service channel access is critical, this bit should be set to (1). The user tasks will suspend any SERCOS transmission of any text strings or lists from the user interface, and the communication error "!78 Service channel in use" will be issued.

If user task service channel access is not critical, parameter lists and oscilloscope are seldom used during normal operation, or nuisance "78" errors occur while viewing parameters, this bit should be set to (0).

Note: *Even with prioritization, service channel access can vary between 10 and 100 ms. Therefore, any time-critical parameter transfers should be from the cyclic data if possible; or should be put into a non-critical section of the user program.*

Bit 13: Ignore Drive Warnings

0 = Drive warnings cause the CLC to issue a shutdown error

1 = The CLC ignores drive warnings

Default: 0

By default, the CLC issues the error 498 Drive D Shutdown Error when the drives issue a Class 2 Diagnostic Warning. Sometimes it is necessary to disable this error to allow warning checks to be done in the user program before the system is shut down, or to prevent nuisance faults from being issued during testing. This is a global option for all axes in the system.

Bit 14: Ignore Axis Ready Status in Program Commands

0= Error is issued if axis is not ready

1= No error is issued if axis is not ready

In some applications, it is necessary to add and remove drives from the

system by setting the disable bit in the axis control register. If this bit is (0) (default), the CLC issues an error if the drive is ready before any commands that start motion. These commands include the start command, the homing command, and the operation mode switch. If this bit is (1), the CLC does not issue an error if the drive is not ready.

Bit 16: Disable AT Timing Check

0 = Check SERCOS AT timing (RECOMMENDED)

1 = Do not check AT timing

Default: 0

An option to disable drive telegram time checking is necessary for older versions of DDS 2.1 SERCOS interface cards (DSS and DSI cards). When Bit 16 in System parameter C-0-0010 is (0) (default, recommended), the AT time check is enabled. When it is (1), the AT time check is disabled. If SERCOS disconnect errors occur with this firmware version and had not with previous versions, the AT time check should be disabled or the DSS card should be upgraded.

C-0-0012 Serial Port B Device Type

Selections:	0=	off
	1=	ASCII Host Protocol
	2=	BTC06 Teach Pendant
Default:	2 (BTC06)	
Attributes:	Integer, Read/write in parameter mode	
	Must Reset CLC for option to take effect	

This parameter selects the type of device that is connected to serial port B. This selection takes effect at the next reset (from power-up or reset switch) of the CLC. The baud rate and options of this port are configured in parameter C-0-0004 Serial Port B Setup.

C-0-0013 Serial Port A Mode

Selections:	232=	RS-232
	422=	RS-422
	485=	RS-485
Default:	232	
Attributes:	Integer, Read/write at any time (on CLC-D or P)	
	Read-only on CLC-V (always 232)	

This parameter selects the serial port A communication mode. This selection takes effect immediately. The baud rate and options of this port are configured in parameter C-0-0003 Serial Port A Setup. On the CLC-V, only RS-232 is allowed.

RS-232

Connect to one device at a time according to RS-232 standard. Maximum cable length is 20 meters.

RS-422

Connect to one device at a time according to RS-422 standard. Maximum cable length is 200 meters.

RS-485

The CLC is a slave on a multi-drop ring with a host and up to 16 slaves, using the RS-485 standard.

Maximum cable length is 200 meters.

C-0-0014 Serial Port B Mode

Selections:	232=	RS-232
	422=	RS-422
	485=	RS-485
Default:	232	
Attributes:	Integer, Read/write at any time	
	(on CLC-D or P)	
	Read-only on CLC-V (always 232)	

This parameter selects the serial port B communication mode. This selection takes effect immediately. The baud rate and options of this port are configured in parameter C-0-0004 Serial Port B Setup. Connection modes include RS-232, RS-422, and RS-485.

C-0-0016 Communication Time-out Period

Units:	millisec
Default:	2000
Attributes:	Integer, Read/write at any time

This parameter adjusts the communication time-out period. The state of the communication error timer is set to enabled/disabled by start/stop commands from the serial device. A *CLC Shutdown Error* occurs if the communication error timer changes states from the enabled to the timed-out state. Only a *Fault Clear* can move the communication timer from the timed-out to the disabled state. If the communication timer is enabled, the timer is reset after each valid Reset Timer Command. The communication timer is reset by both a Timer Reset command and a change of state from disabled to enabled (i.e., Start Timer command).

C-0-0020 Transmitter Fiber Optic Length

Units:	meters
Default:	0.0 = minimum transmit power
Attributes:	Float, Read/write at any time
	(takes effect next shift down in phase)

This parameter adjusts the intensity of the output from the CLC's SERCOS transmitter, based on the length of the cable in meters. The length values are for plastic fiber. For glass fiber, a smaller length must be entered for the equivalent power. In future releases, another parameter will be added to select glass or plastic.

C-0-0021 User Watchdog Timer

Units: msec
Default: 0
Max: 2^32-1
Attributes: Integer, Read/write at any time

The user watchdog timer enforces a time constraint on a user task or a user interface.

Every time a nonzero timeout value is written to C-0-0021, a timer is triggered on the CLC. If the timeout expires, the error "508 User Watchdog Timeout" is issued. The timer is checked by the CLC every 50ms.

If C-0-0021 is set to zero, the watchdog timer is disabled. If it is nonzero, it is active when the CLC is in run mode, there are no errors, and the task specified in C-0-0022 is running.

In a user program task, parameter C-0-0021 can be written to via a parameter transfer at the beginning of the main processing loop. If the CLC system tasks or the user program events are consuming too much processor time, the time set in C-0-0021 will elapse, and error 508 will be issued. The programmer can then adjust the timing of the events, or increase the SERCOS or I-O cycle times to allow more time for the user task.

C-0-0022 User Watchdog Task ID

Units: 0-4
Default: 1
Max: 4
Attributes: Read/write at any time, , default=1, Integer

Parameter C-0-0022 selects the task that must be running in order for the watchdog timer to be active. If it is set to 0, no tasks are associated with the watchdog timer, and it must be updated when the CLC is in run mode.

C-0-0022 value	User watchdog timer in C-0-0021 is active when:
0	The CLC is in Run mode, there are no errors, and C-0-0021 is nonzero.
1	Task A is running, there are no errors, and C-0-0021 is nonzero.
2	Task B is running, there are no errors, and C-0-0021 is nonzero.

3	Task C is running, there are no errors, and C-0-0021 is nonzero.
4	Task D is running, there are no errors, and C-0-0021 is nonzero.

C-0-0030 Option Cards Active

Attributes: Read-only, binary word

Note: Rightmost bit is bit 1.

Bit	Option Card (0=not active, 1=active)
11-16:	Reserved
10:	DPF Profibus Slave Card
9:	DBS Interbus-S Slave Card
4-8:	Reserved
3:	DEA 30 Parallel I-O
2:	DEA 29 Parallel I-O
1:	DEA 28 Parallel I-O

When an option card is found on the CLC-D expansion bus, the bits corresponding to each device are set to (1). Registers, communication buffers, and drivers are automatically configured for the corresponding interface.

Option cards not found or not connected are indicated by (0).

C-0-0031 Option Card Status

Attributes: Read-only, binary word

Note: Rightmost bit is bit 1.

The bits in this parameter indicate the status of each active option card. The bits correspond to the devices in C-0-0030. When an error condition exists on an option card, its bit is set to (1). When an option card is operating properly, its bit is (0).

DEA/C Cards (bits 1-3)

The DEA/C parallel I-O cards monitor the external power supply voltage and the output driver current. If the supply voltage is less than 18V (too low), the bit is set to (1). If the supply voltage is O.K., the bit is set to (0). ***See the description of CLC shutdown error 507 for more information.***

DPF, and DBS Cards

The operation of the status bits corresponding to these cards are not defined in this version.

Jogging and Display (0042-0056)

C-0-0042 World Large Increment

Units: Percent
Limits: 1-99
Default: 50
Attributes: Integer, Read/write at any time

This parameter sets the incremental coordinated jogging distance that is used when Large Increment is selected. It is based on the value entered in task parameter T-0-0025 Maximum Jog Increment.

C-0-0043 World Small Increment

Units: Percent
Limits: 1-99
Default: 50
Attributes: Integer, Read/write at any time

This parameter sets the incremental coordinated jogging distance used when Small Increment is selected. It is based on the value entered in task parameter T-0-0025 Maximum Jog Increment.

C-0-0045 World Fast Jog Speed

Units: Percent
Limits: 1-99
Default: 50
Attributes: Integer, Read/write at any time

This parameter sets the coordinated jogging speed used when Fast Jog is selected. It is based on the value entered in task parameter T-0-0026 Maximum Jog Velocity.

C-0-0046 World Slow Jog Speed

Units: Percent
Limits: 1-99
Default: 50
Attributes: Integer, Read/write at any time

This parameter sets the coordinated jogging speed used when Slow Jog is selected. It is based on the value entered in task parameter T-0-0026 Maximum Jog Velocity.

C-0-0052 Axis Large Increment

Units: Percent
Limits: 1-99
Default: 50

Attributes: Integer, Read/write at any time

This parameter sets the incremental jogging distance used when Large Increment is selected. It is based on the value entered in axis parameter A-0-0025 Maximum Jog Increment.

C-0-0053 Axis Small Increment

Units: Percent
Limits: 1-99
Default: 50
Attributes: Integer, Read/write at any time

This parameter sets the incremental jogging distance used when Small Increment is selected. It is based on the value entered in axis parameter A-0-0025 Maximum Jog Increment.

C-0-0055 Axis Fast Jog Velocity

Units: Percent
Limits: 1-99
Default: 50
Attributes: Integer, Read/write at any time

This parameter sets the axis jogging speed used when Fast Jog is selected. It is based on the value entered in axis parameter A-0-0026 Maximum Jog Velocity.

C-0-0056 Axis Slow Jog Velocity

Units: Percent
Limits: 1-99
Default: 50
Attributes: Integer, Read/write at any time

This parameter sets the axis jogging speed used when Slow Jog is selected. It is based on the value entered in axis parameter A-0-0026 Maximum Jog Velocity.

Program Management (0090-0099)

C-0-0090 Download Block Size

Data: 1-200
Default: 16
Attributes: Integer, Read/write at any time

This parameter selects the block size that will be used for user program

downloads to the CLC. Refer to the section on Direct ASCII Communication for more information.

C-0-0091 Total Program Memory

Units: Bytes

Attributes: Integer, Read-only

This is the total file memory on the CLC card that can be used for programs and text messages. CLC point, variable, I/O, and event tables are included in this number.

C-0-0092 Available Program Memory

Units: Bytes

Attributes: Integer, Read-only

This is the amount of memory the CLC currently has available for storage of programs and text messages.

C-0-0093 Contiguous Program Memory

Units: Bytes

Attributes: Integer, Read-only

The CLC dynamically allocates the non-volatile memory used for user programs and data. This is the largest unit of storage (i.e. the largest program or cam) that can be stored on the card. If this number is much smaller than C-0-0092 Available Program Memory, Available Program Memory, the memory is fragmented. If the message “Insufficient Program Space” is sent from the CLC, upload programs, delete them, then download them again to de-fragment the memory.

C-0-0094 Maximum Executable Program Size

Units: Bytes

Attributes: Integer, Read-only

The CLC reserves a fixed amount of memory to store the currently active executable program. The executable program contains the instructions but not the points, events, variables, and labels. This parameter indicates the largest executable program that can be stored on the CLC. The program size can be checked against this number before the program is downloaded. If the CLC receives a user program that exceeds this number, communication error “!60 Executable program is too large” is issued.

C-0-0095 Path Planner to SERCOS Time Factor

Min: 1

Max: 100

Allows the coordinated motion path planner to run less often (a multiple of

the SERCOS rate) than the SERCOS cycle time to improve system performance.

Example: If the SERCOS rate is set to 4ms and C-0-0095=5 the path planner would run every 5th SERCOS cycle or every 20ms.

This factor can reduce overhead due to path planner calculations by 120% or more. This allows user Tasks A-D to appear to run much faster.

If this factor is too large it can result in a degradation of the coordinated path. Optimal settings for this parameter would be between 3 and 6.

C-0-0098 Initialization Delay

Units:	seconds
Default:	0 (no delay)
Limits:	0-30
Attributes:	Integer, Non-volatile, Read/write in parameter mode

This parameter causes the CLC to delay for the specified number of seconds before it initializes the SERCOS ring. This prevents the CLC from issuing a "No drives were found on ring" error if I-O stations or drives take a long time to initialize the SERCOS ring.

C-0-0099 Minimum SERCOS Cycle Time

Units:	microseconds
Default:	2000 for CLC-GPS
	4000 for CLC-GPR
Attributes:	Integer, Read/write in parameter mode

For some applications, such as coordinated motion using multiple tasks or cam motion using multiple cams, the application program can be made faster by increasing the SERCOS cycle time.

This parameter's operation depends on the hardware and software versions.

CLC-D/P and CLC-V 2.3

with separate SERCOS clocks

The CLC automatically sets the SERCOS cycle time based on the number of drives and the amount of data on the ring. Parameter C-0-0099 sets the minimum SERCOS cycle time. This parameter takes effect when the SERCOS phase is decreased (when switching into parameter mode or activating a program).

CLC-V 2.3

with common SERCOS clock

When a common SERCOS clock through the VME backplane is enabled with CLC C-0-0010 System Options, the cycle time is not automatically configured by the CLC. Parameter C-0-0099 must be set equal for all cards using the same SERCOS clock. The CLC issues an extended diagnostic message when the selected cycle time is too small for the number of drives or the amount of data on the ring. This parameter takes effect when the

SERCOS phase is decreased (when switching into parameter mode or activating a program).

System Status (0100-0126)

System status messages are available through the serial ports and the user program, and provide the current status of the CLC system.

C-0-0100 CLC Firmware Version

Type: String
Attributes: Read-only

The CLC displays the firmware version number for the card, which corresponds to the Indramat standard part number:

```
CLC*DP-GPS-03V10
   |      |    |_Firmware Version
   |           | _Control Type (GPS = general purpose)
   |_Hardware Type
```

C-0-0101 CLC Hardware Version

Type: String
Attributes: Read-only

The CLC displays the hardware version type code:

```
CLC-D 1.1 250233 A03
      | |   |      |_board revision
      | |   |_board part number
      | |_Hardware revision number
      |_Hardware platform (CLC-D, CLC-P, CLC-D)
```

C-0-0102 CLC Version Date

Type: String
Attributes: Read-only

This parameter shows the release date of the CLC firmware:

Mar 29 1996

C-0-0103 Maximum Number of Axes Allowed

Type: Integer
Attributes: Read-only

This indicates the maximum axis number and the maximum number of axes allowed in this software version. It can be used to distinguish between standard and economy GPS versions, which otherwise keep the same version number.

C-0-0111 Data Backup Selection

CLC-D/P: 0 = no selection
1 = save communication parameters to EEPROM
2 = restore comm. parameters from EEPROM
3 = erase comm. parameters from EEPROM

Attributes: Integer, read/write in parameter mode only

Default: 0 at power-up

The communication parameters on the CLC-D and CLC-P can optionally be saved in a 256-byte EEPROM on the card. This allows communication to take place even after the RAM backup battery has failed. The parameters stored in RAM are used unless those stored in EEPROM are retrieved or the battery fails. The jumpers S1 and S2 override both the RAM and EEPROM values to allow service personnel to connect at a known baud rate (9600) and mode (RS-232).

This parameter selects the command that will be performed by parameter

C-0-0112 Data Backup Command. On the CLC-D and CLC-P, this command saves and restores communication parameters to and from EEPROM.

When a (1) is entered in this parameter, the values set in the communication parameters are stored into EEPROM when command C-0-0112 is executed.

When a (2) is entered, the values stored in EEPROM are written to the communication parameters when command C-0-0112 is executed. The restored parameters do not take effect until the power is cycled on the CLC.

A (3) erases the communication parameters from EEPROM when command C-0-0112 is executed. When the battery is lost, the defaults for each parameter are loaded.

Parameters Saved

The following communication parameters can be saved to EEPROM:

C-0-0002 Unit Number

C-0-0003 Serial Port A Setup (includes baud rate)

C-0-0004 Serial Port B Setup (includes baud rate)

C-0-0012 Serial Port B Device Type

C-0-0013 Serial Port A Mode

C-0-0014 Serial Port B Mode

C-0-0112 Data Backup Command

Attributes: Binary, Read/write in parameter mode

Default: 0 at power-up

Note: Bit 1 is the rightmost bit, bit 16 the leftmost.

On the CLC-D and CLC-P, this command parameter saves communication

parameters to EEPROM or restores them from EEPROM, according to the selection in parameter C-0-0111 Data Backup Selection.

Bit 1: Set Command

Bit 2: Enable Command

These bits follow the SERCOS specification for procedure commands. The CLC executes a backup when it sees a (0-1) edge in bits 1 and 2. The backup option must first be selected in parameter C-0-0111. To start the backup, restore, or erase, the following sequence must be used:

Bit 2	Bit 1	Data Backup Command
0	0	Disable Command
1	1	Enable and Start Command (Save or Restore Operation is started)

Procedure for Saving Communication Defaults:

1. Set the communication parameters to the desired values.
2. Set parameter C-0-0111 to (1).
3. Set parameter C-0-0112 to (0000000000000000).
4. Set parameter C-0-0112 to (0000000000000011).

C-0-0120 Operating Mode

Codes: 0= Initializing CLC
1= Parameter Mode
2= Run Mode
Attributes: Integer, Read-only

Provides a code corresponding to the current CLC operating mode.

C-0-0121 SERCOS Communication Phase

Codes: SERCOS Phase (0-4)
Attributes: Integer, Read-only

This is the current phase (0-4) of the CLC SERCOS master. Note that the drives may have switched to a lower phase if there was an error.

C-0-0122 Diagnostic Message

Type: String
Codes: 000-199: Status message
200-399: Warning error
400-599: Shutdown error
Attributes: read-only

This is the current system status code and message. ***See Chapter 8. CLC Diagnostics for listings of status and error codes***.

Example: 412 No drives were found on ring
| |_Message
|_Error code

C-0-0123 Diagnostic Code

Type: Integer
Codes: 000-199: Status message
200-399: Warning error
400-599: Shutdown error
Attributes: read-only

This parameter returns a status code. ***See Chapter 8. CLC Diagnostics for listings of status and error codes.***

C-0-0124 Extended Diagnostic

Attributes: String, read-only

This is a dynamic system message used to provide additional diagnostic information for a status warning or error message (parameter C-0-0122 Diagnostic Message).

C-0-0125 System Timer Value

Units: milliseconds
Attributes: Integer, Read/Write at any time

This is a general-purpose timer that continuously counts milliseconds while the CLC is running. It is a global system variable and can be read into an integer variable to provide timing for a section of a CLC program, or its incremental value can be used to time a process. It is a 31 bit counter with a maximum count is 2,147,483,648 (2^31), after which it rolls over to 0 and continues counting. It can be set to any value by the user program or the user interface.

C-0-0126 Date and Time

Type: String
Attributes: Read/Write any time (write requires SET command)

This is the current time and date, which the CLC uses for Flash identification and the diagnostic log. It also provides a time stamp for parameters when the parameter list is uploaded.

The date is printed Month-Day-Year, and the time is Hour-Minute-Second in 24-hour format (example below is 1:05 PM on November 3).

11-03-1994 13:05:24

The host can set the date at any time. To prevent the date from being

changed inadvertently when a parameter file is downloaded, the CLC requires 'SET' before the new time and date:

>SET 11-04-1994 15:33:00

The CLC/V has a real time clock which keeps time even when the power is off. On the CLC/D and CLC/P, the host device must set the time and date when the CLC is powered on, since there is no real time clock.

Electronic Line Shaft Parameters 1 of 2 (0150-0170)

For C-0-1000 to C-0-1568 See Electronic Line Shaft Parameters 2 of 2

C-0-0150 Master 1 Type

Codes: 0 = ELS disabled
1 = Virtual Master
2 = Real master to each slave
3 = Real master through SERCOS
4 = Follow axis feedback

Attributes: Integer, automatically set by ELS/INIT command at program activation

This provides a status of the type of ELS master. See the descriptions of Electronic Line Shafting and the ELS/INIT command.

C-0-0151 Master 1 Drive Address

Selections: 0 if real master is disabled
1-max_axis, real master encoder address

Attributes: Integer, automatically set by ELS/INIT command at program activation

This parameter is set to the address of the drive in which the encoder for a real master resides. See the descriptions of Electronic Line Shafting and the ELS/INIT command.

C-0-0152 ELS Master Task Assignment

Codes: 0 = No task selected
1 = Task A
2 = Task B
3 = Task C
4 = Task D

Attributes: Integer, automatically set by ELS/INIT command at program activation.

This is the task with which the ELS virtual or real master is associated.

When this task goes into manual mode, the virtual master is ramped to zero velocity using the E-Stop Deceleration (C 1.156). The task assignment is set with the first task (A, B, C, D in order) that includes an ELS/INIT command with a virtual master.

C-0-0153 Virtual Master Programmed Velocity

Units: RPM

Default: 0.0

Attributes: Floating point, read/write at any time

This is the target velocity for the ELS virtual master. A negative value causes motion in the negative direction. It is set using the AXIS/INITIALIZE command or VELOCITY icon in the user program.

C-0-0154 Virtual Master Programmed Acceleration

Units: radians/sec2

Default: E-stop deceleration (C-0-0156 Virtual Master E-Stop Deceleration)

Attributes: Floating point, read/write at any time

This is the acceleration rate for the ELS virtual master. The master accelerates to a greater speed using this parameter. It is set using the AXIS/INITIALIZE command or ACCEL icon in the user program.

C-0-0155 Virtual Master Programmed Deceleration

Units: radians/sec2

Default: E-stop deceleration (C-0-0156 Virtual Master E-Stop Deceleration)

Attributes: Floating point, Read/Write at any time

This is the deceleration rate for the ELS virtual master. The master decelerates to a smaller speed using this parameter. It is set using the ELS/DECEL command or DECEL icon in the user program.

C-0-0156 Virtual Master E-Stop Deceleration

Units: radians/sec2

Default: 200.0

Attributes: Floating point, non-volatile, read/write at any time

The CLC Virtual Master uses this parameter to decelerate to zero velocity at an Emergency Stop, a Shutdown error, or when switching into manual mode without a cycle stop.

C-0-0157 ELS Master Current Position

Units: degrees (modulo 360)

Default: 0.0 at power-up or reset

Attributes: Floating point, Read/Write at any time

This run time status parameter is the current position of the active ELS master (real or virtual) in degrees. It is possible to write a value to this parameter to reset the virtual master position, but care should be taken that all axes are stopped.

C-0-0158 ELS Master Current Velocity

Units: RPM

Attributes: Floating point, read only

This run time status parameter is the current velocity of the CLC Virtual or Real Master in revolutions per minute.

C-0-0159 Master 1 Zero Velocity Window

Units: RPM

Default 1.0

Attributes: Floating point, read/write at any time

This parameter determines when the real master is stopped for diagnostic purposes or for a mode change. When the master's speed is below this value, it is considered at zero velocity.

C-0-0160 Virtual Master Maximum Jog Velocity

Units: RPM

Default 100.0

Attributes: Floating point, read/write at any time

This parameter defines the maximum velocity used for jogging the ELS virtual master. Both the fast and slow axis percent parameters are multiplied by this value and used as the master's jog velocity.

C-0-0161 Master 1 Ratio Input

Units: None

Limits: 1.0 to 1*10^9

Default 1.0

Attributes: Floating point, read/write in parameter mode only

See C-0-0162 Master 1 Ratio Output.

C-0-0162 Master 1 Ratio Output

Units: None

Limits: -1 * 10^9 to 1*10^9

Default 1.0

Attributes: Floating point, read/write in parameter mode only

These parameters define a gear ratio for a real master encoder for ELS or cam axes when Real Master through SERCOS or Follow Axis Feedback options are used. The position difference read from the master encoder is multiplied by (C-0-0162 / C-0-0161 Master 1 Ratio Input). The master position is always rotary with a modulo of 360. A ratio output of "-1" changes the direction of the position readout for the master encoder.

Master Position = (Ratio Output / Ratio Input) * Master Encoder Value

C-0-0164 Master 1 Encoder Type

Attributes: Read/write at any time, integer

Units: Milliseconds

Selections: 0 if not used or primary feedback is used

1 if primary feedback is used

2 if secondary feedback is used.

This option selects the type of master used for the ELS 'Follow Axis Feedback' mode.

The primary encoder or secondary encoder can be used. Options (0) and (1) select the master drive's primary feedback, which is the value read from drive parameter S-0-0051. Option (2) selects the secondary feedback, read from drive parameter S-0-0053.

This parameter is set automatically from the els/init instruction.

C-0-0170 PLS 1 Mask Register

Attributes: Read/write at any time, integer

Limits: 0 = no mask is applied to PLS

1 - 1024 = register to be used for PLS mask

Individual bits of the Programmable Limit Switch may be disabled by setting the bits in an optional mask register. On CLC GPx version 03, this register is selected in parameter C-0-0170. For example, if C-0-0170 is set to 110, and register 110 is set to 1111111111110000, the PLS will control only the first 4 bits of the PLS register. The bits that are masked can be controlled via the I-O mapper or the user program. If C-0-0170 is set to 0, no mask will be applied to the PLS.

VME Bus (0200-0287)

C-0-0200 VME Bus Arbitration Mode

Codes:

Bit 1	Bit 0	Arbitration mode
0	0	Prioritized
0	1	Round-Robin
1	0	Prioritized Round-Robin

Bit 2	FAIR Arbitration
0	disabled
1	enabled

Attributes: Byte, Write-only in Parameter Mode

The CLC/VME can be configured to act as the VME bus Arbiter by enabling the slot 1 System Controller functions (*SW5-8*). The Arbiter supports three standard VME algorithms for bus arbitration: prioritized, round-robin, and prioritized round-robin. Parameter 200 is used to select the desired arbitration mode. This parameter is also used to select if the fair arbitration algorithm will be utilized.

Example:

```
xxxxx101

|    | |_ enables Round-Robin arbitration

|    |_ enables FAIR arbitration algorithm

|_ most significant bit
```

C-0-0201 Bus Release Mode

Codes:

Bit 2	Bit 1	Bit 0	Release Inhibit Time
0	0	0	0.5 microsecond
0	0	1	1 µs
0	1	0	2 µs
0	1	1	4 µs
1	0	0	8 µs
1	0	1	16 µs
1	1	0	32 µs
1	1	1	64 µs

Bit 3	Release on Bus Clear
0	disabled
1	enabled

Bit 4	Release Every Cycle
0	disabled
1	enabled

Attributes: Byte, Write-only in Parameter Mode

As the VME bus master, the CLC provides three optional methods for releasing the bus:

ROR - release on request

RBCLR - release on bus clear

REC - release every cycle

Release on request is always enabled. The Release Inhibit Time may be set to insure a minimum time that the CLC will hold the bus as master.

Example:

```
xxx10101

|  |||_|_ sets the release inhibit time to 16µs

|  ||_ disables release on bus clear

|  |_ enables release every cycle

|_ most significant bit
```

C-0-0202 VME/CLC Interface

Codes:

Bit 1	Bit 0	VME bus access time-out
0	0	64000 µs
0	1	1000 µs
1	0	64 µs
1	1	16 µs

Bit 2	VME read/write error
0	VME bus error not fatal to CLC
1	VME bus error fatal to CLC

Bit 3	VME SYSFAIL error
0	VME SYSFAIL ignored by CLC
1	VME SYSFAIL fatal to CLC

Bit 4	Read-Modify-Write cycles
0	Do not allow external RMW cycles to local RAM
1	External RMW to local RAM allowed

Attributes: Byte, Write-only in Parameter Mode

This parameter specifies how the CLC interfaces with the VME bus. Read-Modify-Write cycles are not permitted as the default. Enabling this function will increase the CLC's access time to its local RAM if another card contends for access.

Example:

```
xxx01010

|  |||||_ sets the VME access time to 64 µs

|  |||_ read/write error not fatal to CLC

|  ||_ SYSFAIL error fatal to CLC

|  |_ RMW cycles allowed from external cards

|_ most significant bit
```

C-0-0203 VME Arbiter Status

Codes:

Bits 1	Bit 0	VME bus request level (SW5-6 & 7)
0	0	BR0
0	1	BR1
1	0	BR2
1	1	BR3

Bit 2	VME Slot 1 controller status (SW5-8)
0	VME bus Slot 1 functions disabled
1	VME bus Slot 1 functions enabled

Attributes: Byte, Read-only

This read only parameter permits reading the settings of the VME bus request level and Slot 1 controller circuit board switch (SW5) settings.

Example:

```
xxxxx010

|    |||_ CLC bus requests use priority level BR2

|    |_ CLC not the VME system controller (Slot 1
     disabled)

|_ most significant bit
```

C-0-0210 Unit 0 Short Address Page

See C-0-0285 Unit 15 Short Address Page for description.

C-0-0211 Unit 0 Access

See C-0-0286 Unit 15 Access for description.

C-0-0212 Unit 0 Window Address Page

See C-0-0287 Unit 15 Window Address Page *for description.*

C-0-0215 Unit 1 Short Address Page

See C-0-0285 Unit 15 Short Address Page for description.

C-0-0216 Unit 1 Access

See C-0-0286 Unit 15 Access for description.

C-0-0217 Unit 1 Window Address Page

See C-0-0287 Unit 15 Window Address Page *for description.*

C-0-0220 Unit 2 Short Address Page

See C-0-0285 Unit 15 Short Address Page for description.

C-0-0221 Unit 2 Access

See C-0-0286 Unit 15 Access for description.

C-0-0222 Unit 2 Window Address Page

See C-0-0287 Unit 15 Window Address Page *for description.*

C-0-0225 Unit 3 Short Address Page

See C-0-0285 Unit 15 Short Address Page for description.

C-0-0226 Unit 3 Access

See C-0-0286 Unit 15 Access for description.

C-0-0227 Unit 3 Window Address Page

See C-0-0287 Unit 15 Window Address Page *for description.*

C-0-0230 Unit 4 Short Address Page

See C-0-0285 Unit 15 Short Address Page for description.

C-0-0231 Unit 4 Access

See C-0-0286 Unit 15 Access for description.

C-0-0232 Unit 4 Window Address Page

See C-0-0287 Unit 15 Window Address Page *for description.*

C-0-0235 Unit 5 Short Address Page

See C-0-0285 Unit 15 Short Address Page for description.

C-0-0236 Unit 5 Access

See C-0-0286 Unit 15 Access for description.

C-0-0237 Unit 5 Window Address Page

See C-0-0287 Unit 15 Window Address Page *for description.*

C-0-0240 Unit 6 Short Address Page

See C-0-0285 Unit 15 Short Address Page for description.

C-0-0241 Unit 6 Access

See C-0-0286 Unit 15 Access for description.

C-0-0242 Unit 6 Window Address Page

See C-0-0287 Unit 15 Window Address Page *for description.*

C-0-0245 Unit 7 Short Address Page

See C-0-0285 Unit 15 Short Address Page for description.

C-0-0246 Unit 7 Access

See C-0-0286 Unit 15 Access for description.

C-0-0247 Unit 7 Window Address Page

See C-0-0287 Unit 15 Window Address Page *for description.*

C-0-0250 Unit 8 Short Address Page

See C-0-0285 Unit 15 Short Address Page for description.

C-0-0251 Unit 8 Access

See C-0-0286 Unit 15 Access for description.

C-0-0252 Unit 8 Window Address Page

See C-0-0287 Unit 15 Window Address Page *for description.*

C-0-0255 Unit 9 Short Address Page

See C-0-0285 Unit 15 Short Address Page for description.

C-0-0256 Unit 9 Access

See C-0-0286 Unit 15 Access for description.

C-0-0257 Unit 9 Window Address Page

See C-0-0287 Unit 15 Window Address Page *for description.*

C-0-0260 Unit 10 Short Address Page

See C-0-0285 Unit 15 Short Address Page for description.

C-0-0261 Unit 10 Access

See C-0-0286 Unit 15 Access for description.

C-0-0262 Unit 10 Window Address Page

See C-0-0287 Unit 15 Window Address Page *for description.*

C-0-0265 Unit 11 Short Address Page

See C-0-0285 Unit 15 Short Address Page for description.

C-0-0266 Unit 11 Access

See C-0-0286 Unit 15 Access for description.

C-0-0267 Unit 11 Window Address Page

See C-0-0287 Unit 15 Window Address Page *for description.*

C-0-0270 Unit 12 Short Address Page

See C-0-0285 Unit 15 Short Address Page for description

C-0-0271 Unit 12 Access

See C-0-0286 Unit 15 Access for description.

C-0-0272 Unit 12 Window Address Page

See C-0-0287 Unit 15 Window Address Page *for description.*

C-0-0275 Unit 13 Short Address Page

See C-0-0285 Unit 15 Short Address Page for description

C-0-0276 Unit 13 Access

See C-0-0286 Unit 15 Access for description.

C-0-0277 Unit 13 Window Address Page

See C-0-0287 Unit 15 Window Address Page *for description.*

C-0-0280 Unit 14 Short Address Page

See C-0-0285 Unit 15 Short Address Page for description

C-0-0281 Unit 14 Access

See C-0-0286 Unit 15 Access for description.

C-0-0282 Unit 14 Window Address Page

See C-0-0287 Unit 15 Window Address Page *for description.*

C-0-0285 Unit 15 Short Address Page

Type: Hex Number
Attributes: Read/Write in Parameter mode
Range: 0x0001 to 0x00FF.

These parameters set the short VME address page for the CLC card. All handshaking through the CLC mailbox structures will be accessed from this page. Mailboxes are realized as offsets into the defined short VME area. There are 256 pages possible, each consisting of 256 bytes.

C-0-0286 Unit 15 Access

Type: Byte
Attributes: Read/Write in Parameter mode

These parameters specify the VME bus address, width and data width for access to the CLC local RAM. The CLC can be accessed with either Standard or Extended addressing. All of the data widths are supported.

Bits 1-0	Data Width
00	D8
01	D16
10	D32

Bit 2	Address Range
0	Standard Address Range (A24)
1	Extended Address Range (A32)

C-0-0287 Unit 15 Window Address Page

Type: Hex Number
Attributes: Read/Write in Parameter mode

These parameters are used to specify the address page where the CLC's local RAM will be located in VME address space. The specific addresses used to access CLC back plane communications and IO registers are calculated using these parameters.

The base address must not be set lower than the size of the CLC memory. The lowest address for a 2 Megabyte CLC card is 0x00200000(0x0020). The lowest address for a 4 Mb card is 0x00400000(0x0040).

BTC06 Teach Pendant (0801-0991)

C-0-0801 Pendant Protection Level 1 Password

Default: 0

Limits: 0-9999

Attributes: Integer, Read/write at any time

This parameter defines a four-digit numeric password that prevents entry into protected menus unless it is entered. If it is 0, the password is disabled.

C-0-0802 Pendant Protection Level 2 Password (Not Currently Used)

C-0-0803 Pendant User Accessible Floats Section

Default: -1

Selections: -1 = all are accessible

0 = none are accessible

n - where n = number of program integers defines range

Attributes: Integer, Read/write at any time

This parameter defines the maximum allowable range for program floats to be user accessible from the BTC06. The operator can view all the program floats, but the operator can only access the program floats, up to number set in this parameter. If the operator needs to change a program float greater than the number in this parameter then the operator can either enter a password or set the pendant level protection bits (System Control Register 1, bits 15 & 16).

If this parameter is set to 0, no floats are accessible by the operator. If it is set to (-1), all floats are accessible.

Example: User Accessible Program Float Section = 10

When the operator selects Table Edit Menu/Float Table Menu, the operator can only access the first ten floats. The programmer is responsible for structuring the program floats properly.

C-0-0804 Pendant User Accessible Integers Section

Default: -1

Selections: -1 = all are accessible
0 = none are accessible
n - where n = number of program integers defines range

Attributes: Integer, Read/write at any time

This parameter defines the maximum allowable range for program integers to be user accessible from the BTC06. The operator can view all the program integers, but the operator can only access the program integers, up to number set in this parameter. If the operator needs to change a program integer greater than the number in this parameter then the operator can either enter a password or set the pendant level protection bits (System Control Register 1, bits 15 & 16).

If this parameter is set to 0, no integers are accessible by the operator. If it is set to (-1), all integers are accessible.

Example: User Accessible Program Integer Section = 10

When the operator selects Table Edit Menu/Integer Table Menu, the operator can only access the first ten integers. The programmer is responsible for structuring the program integers properly.

C-0-0805 Pendant Start of User Accessible Registers

Default: 1

Limits: 1 - maximum I-O register number (512 on CLC-D/P, 2048 on CLC-V)

Attributes: Integer, Read/write at any time

See C-0-0806 Pendant End of User Accessible Registers

C-0-0806 Pendant End of User Accessible Registers

Default: maximum I-O register number (512 on CLC-D/P, 2048 on CLC-V)

Limits: 1 - maximum I-O register number

Attributes: Integer, Read/write at any time

These two parameters (C-0-0805 and C-0-0806) define a range of registers accessible to the operator. The operator can view all the registers, but the operator can only access the registers within this window. If the operator needs to change a bit in a register outside the window of this parameter, then the operator can either enter a password or set the pendant level protection bits (System Control Register 1, bits 15 & 16). When the Register I/O Menu is selected on the BTC06, the first register to be displayed is the number stored in the Start of User Accessible Registers parameter.

C-0-0807 Pendant Password Timeout

Units: seconds

Default: 30 seconds

Limits: 0 (disabled) to 3600 (one hour)

Attributes: Integer, Read/write at any time

This parameter sets a timeout on the BTC06 password. After the password is entered (C-0-0801 Pendant Protection Level 1 Password), the user can enter any screens requiring the password for the time set in this parameter. When a key is pressed, the timer is reset. After the timer expires, the password is again required. If the timeout is set to 0, the password is always required.

Note: If the password is used, the pendant protection level bits (Register 1, bits 15 and 16) do not function.

C-0-0810 BTC06 message and prompt control word

Attributes: 16-bit Binary word, Read/Write Any Time

Default: 0 Inactive (no user input requested/no message displayed on the BTC06)

This parameter is used to display a user task status message in the top two lines of the BTC06 and it can prompt the user for data entry into a variable.

Bit Definitions

Bits 8-1:	N	Variable ID number (0-255)
Bits 11-9:	V	Variable type
	0	Integer Variable
	1	Float Variable
	2	Global Integer Variable
	3	Global Float Variable
Bits 14-12:	T	Task ID
	0	Inactive (no messaging)
	1	Use task A's status message
	2	Use task B's status message
	3	Use task C's status message
	4	Use task D's status message
Bits 16-15:	S	Control and status
	00	Done (status)
	01	New prompt request for input (prompt for data entry) or Data entry abort (when N=0 and V=0)
	10	Operator input active (status)
	11	Operator input error (status)

VisualMotion Programming

1. Use the “Message” icon to create the message to be displayed on the first two lines of the BTC06.
2. Use the VisualMotion “Calculator” icon to build an integer that corresponds to the desired bit settings for parameter C-0-0810.
3. Use the “Parameter Transfer” icon to transfer the integer to parameter 810. The integer will be converted to a 16-bit binary word and processed in this parameter.

Note: The programmer should set up the entire word before writing it to the parameter.

The task’s status/prompt message appears on the first two lines of the BTC06 display. The programmer is responsible for formatting the message and taking care of wrap-around. The BTC06 task keeps track of message changes and updates the display accordingly.

A task’s status/prompt message may contain only one data entry field. Field size and location are determined by the following conventions:

- Data entry fields are indicated by one or more ‘#’ signs. Field size is determined by the number of consecutive ’#’ signs used.
- Data entry fields may contain a decimal sub-field. Example: #####.##
- Searching left to right, the first ’#’ sign found determines the field’s location. Example: “Please enter part #: #####”. The first ’#’ sign (“part #”) would erroneously be used for data entry.
- The absence of ’#’ signs in a prompt message forces a default field of 12 characters at the end of a message.
- In the displayed message, ‘#’ signs are blanked out and the cursor is placed left justified in the field.
- When writing the message in the task, the first line can be terminated by using the ‘~’ character. This makes for easy formatting of a message with a prompt on the second line.

Note: The functionality of the VisualMotion Message Setup Box remains unchanged. A variable can still be displayed in the message by using ‘%s’ in a message. A variable that requires data entry must be entered as a series of ‘#’ signs.

Note: The variable defined by the values in V(bits 11-9) and N (bits 8-1) is displayed in the task status message. If V and N are not defined, a formatting string (consisting of one or more ‘#’ signs) that is entered in the VisualMotion Message Icon definition will be displayed as the ‘#’ sign(s) instead of as the variable.

Control and Status

Prompt for user input

When the BTC06 task sees the new prompt status (01b in bits 15 and 16) it prompts the operator for data entry. As soon as the operator begins to enter a value, the BTC06 sets the status to 10b (in bits 15 and 16) indicating that the operator is in the process of entering numerical data. No additional messages can be placed until the data entry is complete (press of OK or ESC key) or a Data entry abort is commanded.

When the operator presses the OK or ESC key, the BTC06 task sets the status to 00b, Done (in bits 15 and 16), indicating that data input is complete and successful (no input error). The programmer should then check the input against internal maximum and minimum values.

Data entry abort

If desired, the programmer can use the VisualMotion “Parameter Transfer” icon to force an Abort Message by setting a 01b status in bits 15 and 16 with V=0 and N=0. This abort can override the user input when a certain condition is met in the system (e.g. a hazardous condition).

Data input error

When a data input error occurs, the BTC06 flashes the data field for 3 seconds. Bits 15 and 16 are set to 11b indicating the data input error, and the variable remains unchanged.

C-0-0811 User Task Controlled Menu ID for BTC06

Attributes: Integer, Read/Write any time
Default: 0 Inactive
Limits: 0 - 19

This parameter allows the programmer to control or “force” which menus the user can access during a task.

Selections:		
	0	Inactive
	1	Main menu
	2	Program Menu
	3	Table Edit Menu
	4	Absolute Table Edit Menu
	5	Relative Table Edit Menu
	6	Event Table Edit Menu
	7	Integer Table Edit Menu
	8	Float Table Edit Menu
	9	Global Integer Table Edit Menu
	10	Global Float Table Edit Menu
	11	Jog Menu
	12	Control Menu
	13	Register I/O Menu
	14	Parameter Menu
	15	Card Parameter Menu
	16	Task Parameter Menu
	17	Axis Parameter Menu
	18	Drive Parameter Menu
	19	Diagnostic Menu

The BTC06 task continually scans parameter C-0-0811 for a non-zero value.

As long as this parameter remains zero, VisualMotion user tasks have no menu control over the BTC06.However, when this parameter is set to a non-zero value, VisualMotion user tasks have complete control over BTC06 menus.

Writing zero into C-0-0811 relinquishes user task control over BTC06 menus. The BTC06 continues to display the last active menu, but now the operator can freely select menus.

The programmer selects the desired active menu in a user task by writing a menu ID number into C-0-0811. At this point, the user can move only to adjoining menus lower in hierarchy, but not back to the higher menus. For example, if the control menu is “forced,” the user can move from there to the diagnostic menu, but not back to the main menu. (For a map of the menu hierarchy, refer to the Display Map in the CLC Startup Guide.) To prevent movement between menus, the programmer can use Register 92 to mask the functionality of the F-keys.

VisualMotion Programming

For each menu to be “forced” during a task, the programmer writes the corresponding menu ID number into C-0-0811, using the VisualMotion “Parameter Transfer” icon. The BTC06 task knows that a new menu request has been made by seeing the transition of C-0-0811 from its current value to a different value. The transition triggers the menu to change.

Note: A menu that may cause motion, such as the jog menu, checks that the selected task is stopped before motion is allowed. The error: “User task must be stopped” is issued if this is not the case (e.g.jogging is allowed only during manual or auto mode, but not while the . task is running).

Motion menus check whether the task in control has any motion queued to the path planner. For example, if a cycle stop is executed while motion is active, any attempt to jog results in the following error: “User task has motion pending”. To allow jogging, all motion must be cleared from the path planner by switching to manual mode or performing a coordinated abort in the VisualMotion Stop Icon.

IMPORTANT: Parameter C-0-0811 must be active (≠0) for parameters C-0-0812, C-0-0813, and C-0-0814 to be functional.

IMPORTANT: System errors are handled in the same way, regardless of the user’s ability to control menu selection. System errors automatically transfer control to the diagnostic screen.

C-0-0812 User Task Controlled Task ID for BTC06

Attributes: Integer, Read/Write any time

Default: 0 Inactive

Limits: 0 - 3

This parameter allows the programmer to control or “force” which task motion system is displayed on the BTC06 for all axes and instructions defined in the active task. The user can also select the displayed task motion system by choosing it from the task menu. Using parameter C-0-0812, the programmer chooses the task for the user.

0 Task A

1 Task B 2 Task C

3 Task D

Note: Parameter C-0-0811 must be active (≠0) for parameter C-0-0812 to be functional.

C-0-0813 User Task Controlled Axis Number for BTC06

Attributes: Integer, Read/Write any time

Default: 1 Inactive

Limits: 1-n n = Maximum Number of Axes

This parameter allows the programmer to control or “force” which single axis (defined by that axis’ SERCOS address) is to be jogged. (The user can also select the axis by choosing it from the Jog Menu.) Using parameter C-0-0813, the programmer chooses the axis for the user.

If an invalid axis or point number is found, the BTC06 responds by issuing an error message. It is the programmer’s responsibility to ensure these parameters are set appropriately before taking control of a menu.

Note: Parameter C-0-0811 must be active (≠0) for parameter C-0-0813 to be functional.

C-0-0814 BTC06 Data Transaction Word

Attributes: 16-bit Binary Word, Read/Write Any Time

Default: 0 Inactive

Limits: 0 - 511

This parameter allows the programmer to:

- monitor the user's index value in a particular BTC06 menu (regardless of the user's permission to change the field value)
- direct the user's data entry in a menu (e.g. to direct the point to be taught in the Jog Menu, the register number in the Register Menu, the current point number in the Float Table Menu, etc.), with the ability to assign the user read-only ("lock") or write ("set") privileges

Bit Definitions

Bits 09-01 D Data Index (depends on the selected menu ID)

Non-zero data index is cleared on a failed write

Bits 14-10 M Menu ID Number

Menu #	Menu Name	Index
1	Main Menu	not defined
2	Program Menu	line number
3	Table Edit Menu	not defined
4	Absolute Table Menu	point number
5	Relative Table Menu	point number
6	Event Table Menu	event number
7	Integer Table Menu	integer number
8	Float Table Menu	float number
9	Global Integer Menu	global integer number
10	Global Float Menu	global float number
11	Jog Menu ABS[] Menu	point number
12	Control Menu	not defined
13	Register Menu	register number
14	Parameter Table Select Menu	not defined
15	Card Parameter	not defined
16	Task Parameter	not defined
17	Axis Parameter	not defined
18	Drive Parameter	not defined
19	Diagnostic Menu	not defined

Bits 16-15 S Status and Control

00 Done (status)

01 Read Request

10 Set/unlock command (write a value in a specific screen, editable by the user)

11 Lock command (write a value in a specific screen, not editable by the user)

VisualMotion Programming

1. Use the VisualMotion “Calculator” icon to build an integer that corresponds to the desired bit settings for parameter C-0-0814.
2. Use the “Parameter Transfer” icon to transfer the integer to parameter C-0-0814. The integer will be converted to a 16-bit binary word and processed in this parameter.

Note: The programmer should set up the entire word before writing it to the parameter.

The user task builds this word and sets the transaction request in S (bits 15 and 16. The transaction is complete when the Done status (S=00) is set by the BTC06.

Note: If the absolute point number is “locked” in the Jog Menu after teaching a point, the BTC06 does not automatically advance the point number. The point number is always dictated by parameter C-0-0814.

Note: Parameter C-0-0811 must be active (≠0) for parameter C-0-0814 to be functional.

Generic Cases

S=10, M=0 and D=0 clears all locks in all menus

S=10, M>0 and D=0 clears locks in a specific menu

S=11, M=0 and D=0 locks all data indexes in all menus

S=11, M>0 and D>0 locks data index in a specific menu

S=11, M=0 and D>0 locks data index in the current menu

S=10, M>0 and D>0 sets data index in a specific menu

S=10, M=0 and D>0 sets data index in the current menu

S=01, M=0 and D=0 requests data index from current menu the user is viewing

S=01, M>0 and D=0 requests data index from a specific menu

C-0-0990 Exit to Monitor Prompt

Attributes: Read/Write in Parameter Mode only

Default: NO

This parameter stops the CLC firmware and exits to a monitor prompt in order for the CLC-P2 and CLC-V firmware to be updated through the serial port. To exit to the prompt change the "NO" string value to "PROBE".

Most system errors that would cause an exit to the probe prompt are now reported as shutdown error "514 CLC System Error Code #". Information about the last CLC system error is now displayed in C-0-0991.

C-0-0991 System Error Information

This parameter stores the information concerning the last CLC System Error. ***See Error 514 CLC System Error.***

Most system errors that would cause an exit to the probe prompt are now reported as shutdown error "514 CLC System Error Code #". Information about the last CLC system error is now displayed in C-0-0991.

Electronic Line Shaft Parameters 2 of 2 (1000-1568)

For C-0-0150 to C-0-0164 see Electronic Line Shaft Parameters 1 of 2

C-0-1000 ELS Primary Master

Units: none
Selections: 0 , 1, or 2
Default: 0
Attributes: Integer, Read/write in parameter mode only, set automatically by els/init instruction.

This parameter selects the real master to be used as the primary real master for ELS. If it is set to 0, a secondary real master is not used. If it is set to 1, real master 1 is used. If it is set to 2, real master 2 is used.

C-0-1001 ELS Secondary Master

Units: none
Selections: 0 , 1, or 2
Default: 0
Attributes: Integer, Read/write in parameter mode only

This parameter selects the real master to be used as the secondary real master for ELS. If it is set to 0, only a virtual master is used. If it is set to 1, real master 1 is used. If it is set to 2, real master 2 is used.

C-0-1010 Master Switching Threshold

Units: degrees
Limits: 0.00001 to 10000000.0
Default: 0.1
Attributes: Floating Point, Read/write at any time

When the ELS master is switched from virtual to real or from real 1 to real 2, the positions must match before the switch takes place. This parameter defines the window around which the two master positions are considered equal. If abs(master 1 - master 2) < C-0-1010, then master is switched. Setting this parameter to a large value switches the masters immediately.

C-0-1011 Difference Between Real Master Positions

Units: degrees
Attributes: Floating Point, Read only

This parameter displays the position difference between the primary real master and the secondary real master. This can be read in the user program for diagnostic purposes.

C-0-1012 Master Synchronization Acceleration

Units: rad/sec^2

Attributes: floating point, read/write in all modes

This parameter sets the acceleration at which the synchronization between the virtual and the real master takes place. See the description of ELS master synchronization.

C-0-1013 Master Synchronization Time Constant

Units: seconds

Attributes: floating point, read/write in all modes

This parameter sets the time duration of the phase offset move that is performed at the end of the master synchronization procedure.

C-0-1014 Master Synchronization Velocity Window

Units: RPM

Attributes: floating point, read/write in all modes

During virtual-to-real master synchronization, the difference between both master's velocities is monitored. If the difference between the current real master velocity and the real master velocity at the start of synchronization is greater than this window, the CLC shutdown error "510 ELS Master Synchronization Error" is issued. If this parameter is set to 0, velocity monitoring is disabled.

C-0-1015 Virtual Master Current Position

Units: degrees

Attributes: Floating Point, Read/write at any time

This parameter displays the current virtual master position at all times, even while a real master is active.

It is possible to write a value to this parameter to reset the virtual master position, but care should be taken that all axes are stopped or are not synchronized to the virtual master.

C-0-1016 Virtual Master Current Velocity

Units: RPM

Attributes: Floating Point, Read only

This parameter displays the current velocity of the virtual master.

C-0-1017 Position Difference Selection

Attributes: Read/Write at any time

ELS master position differences can be monitored with parameters and register bits. This ensures an accurate position difference and reduces the processing that needs to be done in the user program.

See Parameters C-0-1018,C-0-1019 and Register 30 bits 11 and 12.

0 = position monitoring disabled

1 = difference Virtual Master - Real Master 1

2 = difference Virtual Master - Real Master 2

3 = difference Real Master 1 - Real Master 2

This parameter selects the position difference that is used for the position difference monitoring feature.

For options 1 and 2, the virtual master modulo of 360 is used, and the difference shown in C-0-1019 will be from -180 to +180. For option 3, the modulo of real master 1 is used, if it is a modulo axis. Otherwise, a linear position difference is used.

This Parameter can be written to in Phase 4.

C-0-1018 Position monitoring window

Units: Degrees

Attributes: Read/Write at any time, float

This master defines a window that affects status bits 11 and 12 in ELS-status register 30. ***See Parameters C-0-1017,C-0-1019 and Register 30 bits 11 and 12.***

C-0-1019 Position difference

Units: Degrees

Attributes: Read Only, float

Note: *If option 3 in C-1017 is selected, it will display the same value as C-1011.*

This parameter displays the difference in position between the two position values as selected in C-0-1017.

See Parameters C-0-1017,C-0-1018 and Register 30 bits 11 and 12.

C-0-1508 Master 1 Filter Cutoff Frequency

Units: Hertz

Attributes: Read/write at any time, float

When a filter is chosen for the ELS real master the cutoff frequency for the filter must be entered. The cutoff frequency is the frequency where the signal is reduced 3db. When set to 0 the filter is disabled. (***See PID descriptions - VM Menu Commands and Icon Programming***)

See C-0-1509 Master 1 Filter Type

C-0-1509 Master 1 Filter Type

Attributes: Read/write at any time, integer

In some systems, the master could cause disturbances due to contact with a part. These disturbances cause oscillations in the master encoder signal. To minimize the oscillations, the CLC provides an optional filter.

This parameter selects the type of filter for the ELS real master. These selections are the same as the ones that are available in the CLC PID function. The filters below are described in the PID description. (***See PID descriptions - VM Menu Commands and Icon Programming***)

0= no filter
1= 1st order low pass
2= 2nd order low pass (default if C-0-1508 Master 1 Filter Cutoff Frequency> 0)
3= 3rd order low pass
4= 2nd order Butterworth
5= 3rd order Butterworth
6= 2nd order low pass with velocity feed forward
7= 3rd order low pass with velocity and acceleration feed forward

See C-0-1508 Master 1 Filter Cutoff Frequency.

C-0-1517 Master 1 Current Position

Units: degrees
Attributes: Floating Point, Read/write at any time

This parameter displays the current position of the first real master.

It is possible to write a value to this parameter to reset the master position, but care should be taken that all axes are stopped or are not synchronized to this master.

C-0-1518 Master 1 Current Velocity

Units: RPM
Attributes: Floating Point, Read only

This parameter displays the velocity of the first real master.

C-0-1550 Master 2 Type

Codes: 0= disabled
2= Real master to each slave
3= Real master through SERCOS
4= Follow axis feedback

Attributes: Integer

This provides a status of the type of second real master. See the descriptions of Electronic Line Shafting and the ELS/INIT command.

C-0-1551 Master 2 Drive Address

Selections: 0 - if real master is disabled
1 - max_axis, real master encoder address

Attributes: Integer, automatically set by ELS/INIT command at program activation

This parameter is set to the address of the drive in which the encoder for a real master resides. See the descriptions of Electronic Line Shafting and the ELS/INIT command.

C-0-1552 Master 2 Encoder Type

Attributes: Read/write in parameter mode only, integer

Units: None

Selections: 0 if not used or primary feedback is used
1 if primary feedback is used
2 if secondary feedback is used.

This option selects the type of master used for the ELS 'Follow Axis Feedback' mode. The primary encoder or secondary encoder can be used. Options (0) and (1) select the master drive's primary feedback, which is the value read from drive parameter S-0-0051. Option (2) selects the secondary feedback, read from drive parameter S-0-0053.

C-0-1554 Master 2 Ratio Input

Units: None

Limits: 1.0 to 1*10^9

Default 1.0

Attributes: Floating point, read/write in parameter mode only

See C-0-1555 Master 2 Ratio Output

C-0-1555 Master 2 Ratio Output

Units: None

Limits: -1 * 10^9 to 1*10^9

Default 1.0

Attributes: Floating point, read/write in parameter mode only

These parameters define a gear ratio for the a real master encoder for ELS or cam axes. The position difference read from the master encoder is multiplied by (C-0-1555 / C-0-1554). The master position is always rotary with a modulo of 360. The ratio output can be positive or negative, allowing a direction change independent of the value read from the drive.

Master Position = (Ratio Output / Ratio Input) * Master Encoder Value

C-0-1556 Master 2 Zero Velocity Window

Units: RPM

Default: 1.0

Attributes: Floating point, read/write at any time

This parameter determines when the real master is stopped for the master stopped status bit, the axis/wait command, or a mode change. When the master's speed is below this value, it is considered at zero velocity.

C-0-1558 Master 2 Filter Cutoff Frequency

See C-0-1508 Master 1 Filter Cutoff Frequency for description.

C-0-1559 Master 2 Filter Type

See C-0-1509 Master 1 Filter Type for description.

C-0-1567 Master 2 Current Position

Units: degrees

Attributes: Floating Point, Read/write at any time

This parameter displays the current position of the second real master. It is possible to write a value to this parameter to reset the master position, but care should be taken that all axes are stopped or are not synchronized to this master.

C-0-1568 Master 2 Current Velocity

Units: RPM

Attributes: Floating Point, Read only

This parameter displays the current velocity of the second real master.

System Parameter Lists (2000-3108)

C-0-2000 List of All Parameters

Type: Parameter List

Attributes: Read-only

This is a multiple-step List Parameter that requests a list of all the parameter numbers in this parameter set. The 'D' subclass of the serial port protocol is used to list these parameters.

C-0-2001 List of Required Parameters

Type: Parameter List

Attributes: Read-only

This is a multiple-step List Parameter that requests a list of all the parameters stored in nonvolatile CLC RAM that must be set for proper CLC operation. The 'D' subclass of the serial port protocol is used to list these parameters.

C-0-2010 List of SERCOS Devices

Type: Parameter List

Attributes: Variable-length list, read-only

When the CLC switches into Communication Phase 1 it scans the SERCOS ring for all connected devices, even if they are not selected in the program. This includes drives and I-O stations. The first step (0) of the list is the number of devices found on the ring. Steps (1) to (number of devices) list the addresses of the drives in sequential order.

C-0-2011 List of SERCOS Drives

Type: Parameter List

Attributes: Variable-length list, read-only

This is a list of all SERCOS devices found on the ring that are drives, even if they aren't selected in the program.

C-0-2012 List of SERCOS I-O Stations

Type: Parameter List

Attributes: Variable-length list, read only

This is a list of all SERCOS devices found on the ring that are I-O stations. The CLC automatically allocates registers for up to five I-O stations with twenty modules per station.

C-0-2013 SERCOS I-O Configuration List

Attributes: Variable-length list, read only

The CLC handles SERCOS I-O from DDS drives and SERCOS I-O stations. DDS drive I-O is configured in axis parameters A-0-0011 Drive I/O Card 1 Setup, A-0-0014 Drive I/O Card 2 Setup, and A-0-0017 Drive I/O Card 3 Setup. SERCOS I-O stations such as Lutze DIOFACE are automatically configured, and no parameters need to be set (see SERCOS I-O and Lutze documentation).

Each step of this list presents the SERCOS I-O configuration in the following format:

Register SERCOS_address Parameter_IDN Direction Number_of_registers

>CD 1.2013.3 100 11 2066 I 1

| | | | | |_number of registers (1, 2, 3, or 4)
| | | | |_'I'= input, 'O'= output
| | | |_SERCOS IDN (corresponds to card or module number)
| | |_SERCOS device address
|_Register number

Example:

```
>  CD 1.2013.0 4 $E1                ;4 elements in list
>  CD 1.2013.1 40 1 32850 I 1       ;drive 1 DEA 4.1 input at register 40, one register
>  CD 1.2013.2 41 1 32849 O 1       ;drive 1 DEA 4.1 output at register 41, one register
>  CD 1.2013.3 100 11 2066 I 1      ;I-O station 11 module 0, input register 100, one register
>  CD 1.2013.4 102 11 2086 O 1      ;I-O station 11 module 2, output register 102, one register
>  CD 1.2013.5 !19 List is finished ;end of list
```

C-0-2014 Start of SERCOS I-O Station Registers

The CLC supports I-O stations that conform to the SERCOS I-O specification, including the Lutze Dioface. CLC GPS-06 can support up to 8 SERCOS I-O stations. A variety of digital and analog I-O Modules can reside on each I-O station.

C-0-2014 selects the starting register number for the group of registers that are to be allocated to the SERCOS I-O Stations. If this parameter value is non-zero, the CLC detects the I-O stations on the SERCOS ring and automatically configures them based on the modules that are present. A parameter list (C-0-0213) of configuration strings indicates the I-O modules that were found and the registers that they are mapped to.

Addresses of I-O Slaves: On the Lutze Dioface, the SERCOS address of an I-O station is selected with 8 binary switches located on the bottom of the SERCOS module. The address can be set to any number that does not conflict with any other drive or I-O station on the SERCOS ring. A list of I-O stations found on the ring is provided in the CLC system parameters.

I-O Station Configuration: The configuration parameters for each I-O module reside at the I-O station, and are communicated via the SERCOS service channel. A parameter list for each module on the station is provided through SERCOS. Refer to the Lutze DIOFACE Manual or SERCOS I-O Specification. Digital I-O will usually work without additional

parameterization. Analog modules usually have scaling parameters associated with them which are stored on the I-O station.

Register Allocation: The CLC reserves registers starting at parameter C-0-2014 for real-time input and output from SERCOS I-O stations. It automatically detects the modules that exist on each station and allocates registers accordingly. Parameter C-0-2015 registers are reserved for up to eight I-O stations found on the ring.

The registers for each station are mapped according to the selected module number. On the Lutze Dioface, the module number can be from 0-63, and is selected with binary switches on the bottom of each module. 32-bit digital I-O modules require two registers and one SERCOS IDN.

Analog I-O modules require 4 registers and 4 SERCOS IDNs. The table below shows the mapping between registers, stations, and modules, and the corresponding SERCOS IDNs.

The module number for a 32-bit register must be an even number, since two 16-bit words are used. Note that on the CLC, only module numbers from 0 to (parameter C-0-2015) can be used. Any other modules that are present will not be mapped to registers.

Register Allocation for SERCOS I-O Stations

Base = base register defined in parameter C-0-2014

Incr = registers per each station defined in parameter C-0-2015

Index = Index of the I-O station found in numerical order.

If SERCOS addresses 4 and 5 are I-O stations, I-O station with SERCOS address 4 would have an Index of 0, and address 5 would have an index of 1.

Register Number	Module Number	SERCOS IDN
Base + (Incr * Index)	0	2066
Base + (Incr * Index) + 1	1	2076
Base + (Incr * Index) + 2	2	2086
Base + (Incr * Index) + 3	3	2096
Base + (Incr * Index) + 4	4	2106
Base + (Incr * Index) + 5	5	2116
Base + (Incr * Index) + 6	6	2126
Base + (Incr * Index) + 7	7	2136
Base + (Incr * Index) + 8	8	2146
Base + (Incr * Index) + 9	9	2156
Base + (Incr * Index) + 10	10	2166
Base + (Incr * Index) + 11	11	2176
Base + (Incr * Index) + 12	12	2186
Base + (Incr * Index) + 13	13	2196
Base + (Incr * Index) + 14	14	2206

Base + (Incr * Index) + 15	15	2216
Base + (Incr * Index) + 16	16	2226
Base + (Incr * Index) + 17	17	2236
Base + (Incr * Index) + 18	18	2246
Base + (Incr * Index) + 19	19	2256

Example:

Base = 400, Incr = 10, and 4 I-O stations (SERCOS addresses 5, 6, 10, 11) were found on the ring.

Registers 400-409 would be allocated to I-O station with address 5.

Registers 410-419 would be allocated to I-O station with address 6.

Registers 420-429 would be allocated to I-O station with address 10.

Registers 430-439 would be allocated to I-O station with address 11.

Module number 2 exists on SERCOS I-O station 6; register 412 is assigned to it. ***Refer to Parameter C-0-2015***.

C-0-2015 Registers Allocated per I-O Station

Selects the number of registers (1-20) used for each I-O Station. These registers are used for real-time input and output from up to 8 SERCOS I-O Stations found on the ring.

C-0-2020 Diagnostic Log List

Attributes: Variable-length list (from 0-100) of strings, read only

The CLC keeps a log of the last 100 shutdown errors. Each diagnostic log string includes the date and time that the error occurred, the shutdown error code, and any other error codes. The most recent error is listed first. The list is retained in battery-backed RAM.

The extended error code is the data that varies in a message, such as the drive number in “Drive D Shutdown Error”. If there is a task or drive error, it is printed as a secondary error code. Error codes are not supported on all versions of drives.

Emergency Stop and warnings are normally not included in this list. To log these errors, set options in parameter C-0-2021 Diagnostic Log Options.

Format:

```
> CD 1.2020.1 11-03 12:15:47 420 1  28 0
                    |     |        |   |   | |_extended secondary error code
                    |     |        |   |   |_secondary error code (28=excessive deviation)
                    |     |        |   |extended error code (1= drive 1)
                    |     |        |_error code (420= Drive D shutdown error)
                    |     |_time that error occurred
                    |_date that error occurred
```

Example:

```
>  CD 1.2020.0 16 $B0                          ;16 errors total
>  CD 1.2020.1 11-03 12:09:03 400 0 0 0        ;Last error was E-Stop
>  CD 1.2020.2 11-01 08:58:59 422 65 6 0       ;next to last was Param
                                                Transfer Task A, code 06
.
.
> CD 1.2020.16 !19 List is finished            ;end of list
```

Emergency Stop and warnings are normally not included in this list. To log these errors, set options in parameter C-0-2021 Diagnostic Log Options.

C-0-2021 Diagnostic Log Options

Format:	Binary Word
Storage:	non-volatile, Read/write in parameter mode
Default:	0000000000000100
Note:	Bit 1 is the rightmost bit.

This parameter sets options for the diagnostic log list in parameter C-0-2020 Diagnostic Log List.

Bit	
1	Include Emergency Stop in the diagnostic log
2	Include Warnings in the diagnostic log
3	Include Drive undervoltage in diagnostic log

By default, the Emergency Stop error (code 400) is not stored in the diagnostic log, since an E-Stop is usually part of normal operation. To store E-stop errors in the diagnostic log, set bit 1 to (1).

To store warnings (codes 200-399) in the diagnostic log, set bit 2 to (1). To exclude warnings, set bit 2 to (0). To store drive undervoltage (drive error 26) in the diagnostic log, set bit 3 to (1). To exclude drive undervoltage, set bit 3 to (0). This bit defaults to (1).

C-0-2501 Oscilloscope signal 1 type

See C-0-2503 Oscilloscope signal 3 type description.

C-0-2502 Oscilloscope signal 2 type

See C-0-2503 Oscilloscope signal 3 type description.

C-0-2503 Oscilloscope signal 3 type

Where signal type is:
0=none.
1=program float.
2=program integer.
3=global float.
4=global integer.
5=axis parameter.
6=unsigned register.
7=signed register.

C-0-2504 Oscilloscope signal 1 id number

See C-0-2506 Oscilloscope signal 3 id number *description.*

C-0-2505 Oscilloscope signal 2 id number

See C-0-2506 Oscilloscope signal 3 id number *description.*

C-0-2506 Oscilloscope signal 3 id number

Number, variable number, parameter number, or register number.

C-0-2507 Oscilloscope signal 1 axis number

See C-0-2509 Oscilloscope signal 3 axis description

C-0-2508 Oscilloscope signal 2 axis number

See C-0-2509 Oscilloscope signal 3 axis description

C-0-2509 Oscilloscope signal 3 axis number

Axis number if signal type is “axis parameter”(1-40) or bit number if signal type is “unsigned register”(1-16).

C-0-2510 Oscilloscope sampling rate

Multiples of SERCOS Cycle time, maximum of 64.

C-0-2511 Oscilloscope signal 1 list

See C-0-2513 Oscilloscope signal 3 list for description.

C-0-2512 Oscilloscope signal 2 list

See C-0-2513 Oscilloscope signal 3 list for description.

C-0-2513 Oscilloscope signal 3 list

These are the lists of captured data, range 1-512 samples.

C-0-2514 Oscilloscope sample count

This sets the sample count in C-0-2511, C-0-2512 and *C-0-2513 Oscilloscope signal 3 list*, range 1 to 512.

C-0-2515 Oscilloscope trigger post-count

Range 1 to sample count. Count equal to sample count (C-0-2514 Oscilloscope sample count) for no pre-trigger.

C-0-2516 Oscilloscope trigger type

Same as signal type above (C-0-2501 Oscilloscope signal 1 type).

C-0-2517 Oscilloscope trigger id number

Variable number, parameter number, or register number.

C-0-2518 Oscilloscope trigger axis or mask

Axis number if signal type is “axis parameter”, or bit number if signal type is “unsigned register”(1-16).

C-0-2519 Oscilloscope trigger level or mask

The value that is compared to C-0-2516 Oscilloscope trigger type, C-0-2517 Oscilloscope trigger id number and C-0-2518 Oscilloscope trigger axis for the internal trigger.

C-0-2520 Oscilloscope trigger mode

1=in positive direction, 2=in negative direction, 3= either positive or negative direction.

C-0-2521 Oscilloscope trigger source

1= user triggered, 2 = internal trigger.

C-0-2522 Oscilloscope trigger control word

Bit 0 - Start recording.

Bit 1 - Trigger start.

Bit 2 - Enable (0->1 Allocates memory and configures system , 1->0 frees memory and disables system).

C-0-2523 Oscilloscope trigger status word

Bit 0 - Trigger activated.

Bit 1 - Data acquisition in progress.

Bit 2 - System Ready(1 indicates memory allocated and system ready, 0 not.

Bit 3 - Capture complete(1 indicates the requested sample count is captured

C-0-2600 Field Bus Object Mapper

Min: 0
Max: 160

Card parameter #2600 is used to configure the cyclic data channel:
CD 0.2600.<l_number> Destination = Source \ n
C-0-2600 will be updated cyclically, while the Mapper of the parameter C-0-2700 will be updated only on the requests of the COM read/write.

C-0-2610 Interbus-S Card Enable

The CLC card parameter C-0-2610 is used to Enable/ Disable the mapping between CLC and dual-port RAM. When CLC powers up, it set to default, 0x80, which enables the mapping.

C-0-2630 Profibus Slave Device Address

Default: 99
Range: 2 to 255

Read/Write DPR Address: 0x5017CC

C-0-2631 Length of Profibus Parameter Channel

Default: 0 - no parameter channel
Range: 0 to 12
Units: bytes

Read/Write DPR address: 0x5017C8

C-0-2632 Profibus Multiplex Control Word

Default: 0xc000
Range: 0 - 0xcfff

Bits 1-16:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bit 15 is bus input enable (1=enable, 0=disable). Bit 14 is bus output enable. Bits 11-8 = the highest multiplex index. Bits 4-0 = multiplexing start offset in bus configuration list (0-28).

C-0-2633 Reserved for future use

C-0-2634 Profibus PD Channel Length

Default: 0
Range: 0 to 100
Units: bytes

Read/Write DPR address: 0x5017CA

C-0-2640 to 2698 Profibus PD Objects

This list contains a series of objects. Some objects represent a 16 bit word, and some of them correspond to a 32 bit double word. Each object can be accessed by a unique physical DPR address. Their numerical values are displayed as integer Hex values (0x0000).

C-0-2700 Field Bus COM Object Mapper

Min: 0
Max: 180

This parameter is used to configure the acyclic data channel. All the objects of the dual-port RAM can be accessed through COM read/write functions. These objects have to be configured in this parameter in order to be directly accessed through the COM channel. The data can be accessed indirectly through the COM channel by using ASCII Communication protocol.

Example: In order to directly read the feedback position of Axis #3, the feedback position should be configured in the Field Bus Mapper.

"> CD 0.2700.1 BO5E98 = AP3.102", where the 32-bit Object 5E98 is used.

After configuration, reading Object 5F98 through COM channel at phase 4 will give the feedback position of the Axis #3.

#2600 will be updated cyclically, while the Mapper of the parameter #2700 will be updated only on the requests of the COM read/write.

C-0-3000 I-O Mapper Program

Type: Parameter List
Attributes: Write in parameter mode only

This variable length list is used to map system control and status to the pre-defined I-O devices. At the start of the list, the number of Boolean strings is sent to/from the CLC. This program is handled as a standard parameter list. If a count of zero is sent, the mapping is deleted. ***See the Chapter 2. CLC I/O Systems for more information.***

Example of sending mapping parameter list:

Host tells the CLC that there are 3 Boolean Strings, and CLC responds:

>1 CP 1.3000.0 3

Host sends the first string:

>1 CP 1.3000.1 120-1 = !1-5

Host closes list after the last string and CLC acknowledges:

>1 CP 1.3000.4 0

>1 CP 1.3000.4 !19 List is finished

C-0-3001 I-O Mapper Options

Default: 0000000000000001

Attributes: Binary Word, Write in parameter mode only

This parameter selects I-O mapper compilation options. Changing these options causes the CLC to recompile the object code.

Bit

1-2 Forcing Options

3-4 reserved

5 Scan Time (!4ms/8ms)

Bits 1-2: Forcing Options

bit2 bit1

0 0 = Disable forcing

0 1 = Force only the reserved control registers (default)

1 1 = Force all registers

I-O forcing allows the results of mapper equations to be ignored if forcing is enabled. Forcing can cause the I-O mapper to execute twice as slow. As options, the CLC can disable forcing, enable forcing for all I-O registers, or enable forcing only for the reserved control registers. Control registers include System control register 1, Axis control register 11, etc. Note: If the BTC06 is used, forcing of the control registers must be enabled.

Bit 5: Scan Time

The I-O mapper on the CLC can be run at either 4 or 8 millisecond scan rates. This allows more strings to be executed with less time taken from user programs if higher-speed I-O isn't necessary.

C-0-3003 I-O Mapper Total Operations

Attributes: Integer, Read Only

This is the total number of operations used by the I-O mapper currently executing on the CLC. Each operand in a Boolean equation or each contact in a ladder diagram corresponds to one operation.

The maximum number of operations allowed on the CLC can be printed using the 'H' (upper limit) subclass of the serial protocol.

C-0-3004 I-O Mapper File Size

Attributes: Integer, Read Only

This is the space consumed in bytes by the I-O mapper text strings on the CLC. The maximum I-O mapper file size allowed on the CLC can be printed using the 'H' (upper limit) subclass of the serial protocol.

C-0-3005 I-O Mapper Executable Size

Attributes: Integer, Read Only

This is the space used by the compiled I-O mapper currently running on the CLC.

The maximum executable size of the I-O mapper can be printed using the 'H' (upper limit) subclass of the serial protocol.

C-0-3100 Cam Tags

Type: Parameter List

Attributes: Read-write at any time if cam is not active

This variable length list allows cams to be named with up to 20 characters. The name can then be displayed or changed by the user interface to identify the cam table or the motion for which it is associated. Each step of the list corresponds to a cam number.

Example of sending cam tags list:

Host tells the CLC that there are 16 names, and CLC responds:

>1 CP 1.3100.0 16

Host sends name for Cam 1:

>1 CP 1.3100.1 cam one

Host closes list after the last string and CLC acknowledges:

>1 CP 1.3100.4 17

>1 CP 1.3100.4 !19 List is finished

C-0-3101 Cam Table 1

See C-0-3108 Cam Table 8 *for description.*

C-0-3102 Cam Table 2

See C-0-3108 Cam Table 8 *for description.*

C-0-3103 Cam Table 3

See C-0-3108 Cam Table 8 *for description.*

C-0-3104 Cam Table 4

See C-0-3108 Cam Table 8 *for description.*

C-0-3105 Cam Table 5

See C-0-3108 Cam Table 8 *for description.*

C-0-3106 Cam Table 6

See C-0-3108 Cam Table 8 *for description.*

C-0-3107 Cam Table 7

See C-0-3108 Cam Table 8 *for description.*

C-0-3108 Cam Table 8

Type: Parameter List

Attributes: Read-write at any time if cam is not active

This variable length list is used to store and retrieve a cam table file. The list starts with step 0 set to the number of entries in the table to be sent to/from the CLC. The rows of the table are then sent as degrees of master, degrees of output, as they would be stored as rows in a space-delineated spreadsheet file.

If a count of zero is sent, the cam is deleted. Otherwise, the count can be from 10 to 1024 table entries. At the end of the download, the CLC expands the table and stores it as 1024 points for accuracy and efficiency. When a table is uploaded from the CLC, the expanded table as stored on the card is sent to the host.

Cam tables may be stored or deleted at any time as long as the cam table number is not active. If the cam is already active, the CLC responds with a communication error “Cam is already active for axis ‘x’”. To download a new table, either switch into parameter mode or deactivate the cam for all axes using it.

Example of sending cam parameter list:

Host tells the CLC that there are 100 table entries:

```
>1 CD 1.3101.0 100          ;100 entries
>1 CD 1.3101.0              ;CLC acknowledgment
```

Host sends the first two entries:

>1 CD 1.3101.1 0.0 0.0	;space delineated x,y of table (x1,y1) = (0.0, 0.0)
>1 CD 1.3101.1	;CLC acknowledgment
>1 CD 1.3101.2 2.0 3.5	;(x2,y2) = (2.0, 3.5)
>1 CD 1.3101.2	;CLC acknowledgment

Host continues for 100 entries, then closes list:

>1 CD 1.3101.101 0.0 0.0	;n+1 closes list
>1 CD 1.3101.101 !19 List is finished	;CLC acknowledgment

3.5 Task Parameters

Class T, Sets A, B, C, D

Each task has a set of parameters that selects options and displays status information. Sets A through D correspond to CLC tasks A through D. These parameters set up and display information for tasks and coordinated motion.

Task Setup (0001-0002)

T-0-0001 Task Motion Type

Codes: 0= No coordinated motion
1= Normal coordinated motion
2= Minimum-time coordinated motion
3= Constant velocity coordinated motion

Attributes: Integer, Read-only

This parameter is set automatically by the user program TASK and PATH commands, and provides the type of task or coordinated motion. Until a valid program is activated, all tasks are non-coordinated by default.

T-0-0002 Task Options

Format: Binary Word

Storage: Non-volatile, Read/write in parameter mode or at program activation.

Default: 0000000000000000 - *Note that bit 1 is the rightmost bit.*

Several options can be selected for the CLC User Tasks A-D. These options are selected via bits in this parameter for each task.

Bit	
1	Run task during all errors
2	Task errors do not shut down other tasks
3	Parameter transfer errors are warnings
4	Automatically start tasks
5	Ignore task and drive errors from other tasks

Bit 1: Run Task During Errors

Some CLC User Tasks are used for motion, and others for communication or I-O. The motion tasks need to be shut down during E-stop, drive, or axis errors to prevent damage or injury. The communication tasks, however, need to convey diagnostic information to an external device at all times. These tasks cannot be shut down when an error occurs.

If this bit is (0), the task will be shut down by any error, according to the other options in this parameter.

If it is (1), the task will run during errors unless a task error is issued in this task. System errors, drive errors, or task errors from other tasks do not stop the task or cause it to execute the error handler.

See the explanation of the Types of Errors at the end of this parameter description.

WARNING: This option should not be selected for a task which has axes associated with it, since the drives will not be shut down during system errors such as E-stop. Option bit 5 in this parameter is a safer method of ignoring drive errors from other tasks.

Bit 2: Task Errors do not Shut Down other Tasks

This option allows errors that occur during the operation of a task to be ignored by other tasks. This option can be used to make motion tasks independent of a communication task.

If this bit is (0), task errors issued from this task will shut down all other user tasks.

If it is (1), all other tasks will continue running during errors issued from this task. Parameter transfer errors and other non-fatal errors will not stop the motion in the other tasks. Drive errors from drives associated with this task are not ignored. To ignore drive errors, it is necessary to enable the option “Ignore task and drive errors from other tasks” (bit 5) for each task.

See the explanation of the Types of Errors at the end of this parameter description.

Bit 3: Parameter Transfer Errors are Warnings

This option allows the current task to continue running if a parameter transfer error occurs. The message “205 Parameter Transfer Warning: see Task A diag.” is issued. If a parameter value is critical to the operation of the task, the task error bit can be tested after the parameter transfer, or this option should be disabled.

Bit 4: Automatically Start Tasks

This option automatically starts tasks and keeps them running. All task control bits are ignored. At exit from parameter mode or when an error is cleared, the task is in placed into automatic mode and started. The task is stopped when parameter mode is selected. This option can be used for supervisory or communications tasks, or to allow the system to start at power-up without any operator intervention.

The Override_Auto_Start bit, bit 2 in the Task Control register, can be used to temporarily disable this function. The bits in the control register are enabled as long as bit 2 is high (1), and are ignored when the bit is low (0). This allows the automatic start option to be disabled in case the task needs to be stopped or the debugging bits need to be used.

Bit 5: Ignore task and drive errors from other tasks

When this option is enabled, errors related to this task's program logic or motion will not affect any other tasks. This option can be used to allow each task to run an independent subsystem of a machine.

If Bit 5 is 0:

All task errors and drive errors will shut down this task. The task stops running, and motion is stopped according to the severity of the error, and the CLC and drive error reaction modes.

If Bit 5 is 1:

This task will continue running during:

1. Task errors issued by other tasks
2. Drive errors from drives associated with other tasks.

This task will be shut down for:

1. System errors such as E-Stop or SERCOS disconnect.
2. Task errors issued by this task.
3. Drive errors from drives associated with this task.

See the explanation of Types of Errors at the end of this parameter description.

Bits 6-16: Reserved for future use

Types of Errors

System Errors include errors that affect the entire CLC system: "400 Emergency Stop" is a system error that shuts down all tasks and all drives. "409 SERCOS Disconnect Error" prevents communication with all drives. Other system errors include internal CLC operating system errors.

Task errors include any error that was issued as the result of a user program instruction. These errors include: parameter transfer errors, errors in the Calc instruction, bounds checking, or motion limit errors related to the instruction, etc. Task errors always set the Task Error bit in the task status register.

Drive errors are issued by the drive, and the CLC displays the error "420 Drive D Shutdown Error". Drive errors set the Class 1 Shutdown bit in the axis status register, but do not set the Task Error bit.

Coordinated Motion (0005-0026)

T-0-0005 World Position Units

Selections:	0= inches
	1= millimeters
	2= radians
Default:	0
Attributes:	Integer, Write in Parameter Mode only

This parameter selects the display units for coordinated motion position, speed, and acceleration data. No unit conversions are performed when changing this parameter. This parameter automatically sets the Axis Position Units parameter (A-0-0005 Linear Position Units) for all coordinated axes in the specified task.

T-0-0010 Kinematic Number

Attributes: Integer, Read-only

The Kinematic Number represents a library routine which identifies the kinematic to be used for coordinated motion. Kinematic routines are application specific and unique to hardware configurations. Consult your Indramat sales office for kinematics to drive your hardware.

T-0-0011 Coordinated X Axis

Codes: 20 if no axis
1 to maximum axis number
Motion Type: Coordinated
Attributes: Integer, Read-only

This parameter provides the axis number (drive number) corresponding to the X axis of this task (selected in the user program TASK/AXIS command). If the number is zero, no coordinated axes are assigned to this task.

T-0-0012 Coordinated Y Axis

Codes: 0 if no axis
1 to maximum axis number
Motion Type: Coordinated
Attributes: Integer, Read-only

This parameter provides the axis number (drive number) corresponding to the Y axis of this task (selected in the user program TASK/AXIS command). If the number is zero, only one coordinated axis is assigned to this task.

T-0-0013 Coordinated Z Axis

Codes: 0 if no axis
1 to maximum axis number
Motion Type: Coordinated
Attributes: Integer, Read-only

This parameter provides the axis number (drive number) corresponding to the Z axis of this task (selected in the user program TASK/AXIS command). If it is zero, only two coordinated axes are assigned to this task.

T-0-0020 Maximum Path Speed

Units: World speed (units/min)
Default: 1000.0
Motion Type: Coordinated
Attributes: Floating point, Write in Parameter Mode Only

This is the maximum speed allowed for this task's coordinated motion. The speed entries in the CLC's Absolute and Relative point tables are percentages of this value.

T-0-0021 Maximum Acceleration

Units: World Accel (units/sec^2)
Default: 200.0
Motion Type: Coordinated
Attributes: Floating point, Write in Parameter Mode Only

This is the maximum acceleration allowed for this task's coordinated motion. The acceleration entries in the CLC's Absolute and Relative point tables are percentages of this value.

T-0-0022 Maximum Deceleration

Units: World accel (units/sec^2)
Default: 200.0
Motion Type: Coordinated
Attributes: Floating point, Write in Parameter Mode Only

This is the maximum deceleration allowed for this task's coordinated motion. The deceleration entries in the CLC's Absolute and Relative point tables are percentages of this value.

T-0-0023 Look Ahead Distance

Units: World position
Default: 10
Motion Type: Coordinated
Attributes: Floating point, Read/write at any time

This parameter sets the minimum look ahead distance that the CLC's path planner uses to calculate a path. The difference between the current target position and the actual position is called the Look Ahead Distance.

The look-ahead distance should never be set to zero. Generally, the look-ahead distance should be twice the length of the longest blend distance. If all blend distances are zero, the look-ahead should be equal to the shortest geometry segment.

Setting the look-ahead to a large value can improve the overall system performance. The length of the look-ahead determines how many intermediate positions the path planner will calculate ahead of the currently commanded position. Of course, if the parameter is set too large the path planner may process many statements before the physical motion takes place. The path planner can't know about potential real-time events that may stop motion or require a program branch, this can result in much wasted calculation and CLC resources.

Decreasing the look-ahead value causes the path planner to process fewer coordinated motion program statements ahead of the current commanded position. This results in a lower potential for wasted calculations and a lighter load on CLC resources. However; motion geometry segments using blending may be missed if the look-ahead value is set too small.

T-0-0024 Velocity Override

Units:	percent
Default:	100.0 (disabled)
Motion Types:	All
Attributes:	Floating point, Read/write at any time

The velocity override provides a method to slow down all motion in a task equally. When a Velocity Override factor is specified for coordinated motion, all velocities in the point table are multiplied by this factor as they are used. When a Velocity Override is specified for non-coordinated motion that is generated by the DDS drive, each velocity command is multiplied by this factor before the command is executed.

T-0-0025 Maximum Jog Increment

Units:	World Position
Default:	1.0
Attributes:	Floating point, Read/write at any time

This parameter defines the maximum distance that is used for incremental coordinated jogging. The Large Increment and Small Increment percent parameters are based on this value.

T-0-0026 Maximum Jog Velocity

Units:	World Velocity (units per minute)
Default:	100.0
Attributes:	Floating point, Read/write at any time

This parameter defines the maximum velocity used for coordinated jogging. The Fast and Slow percent parameters are based on this value.

Robotics (0035-0059)

T-0-0035 Relative Point Used for Origin

Units: none
Default: 0 (default origin is used)
Attributes: Integer, Non-volatile, Read/write at any time

This is the relative point in which the robot origin is stored. The point index in the REL table is stored here at the execution of robot/origin instruction so that the origin stays the same when jogging in manual mode. It can also be edited directly in parameter mode so that the origin can be set up before the execution of the instruction. **NOTE:** *Takes effect at robot/origin instruction execution and at exit from parameter mode.*

T-0-0036 Relative Point Used for Tool Frame

Units: none
Default: 0 (default origin is used)
Attributes: Integer, Non-volatile, Read/write at any time

This is the relative point in which the robot tool frame is stored. The point index in the REL table is stored here at the execution of robot/tool instruction so that the origin stays the same when jogging in manual mode. It can also be edited directly in parameter mode so that the tool frame can be set up before the execution of the instruction. **NOTE:** *Takes effect at robot/tool instruction execution and at exit from parameter mode.*

T-0-0050 Kinematic Value 1

See T-0-0059 Kinematic Value 10 for description.

T-0-0051 Kinematic Value 2

See T-0-0059 Kinematic Value 10 for description.

T-0-0052 Kinematic Value 3

See T-0-0059 Kinematic Value 10 for description.

T-0-0053 Kinematic Value 4

See T-0-0059 Kinematic Value 10 for description.

T-0-0054 Kinematic Value 5

See T-0-0059 Kinematic Value 10 for description.

T-0-0055 Kinematic Value 6

See T-0-0059 Kinematic Value 10 for description.

T-0-0056 Kinematic Value 7

See T-0-0059 Kinematic Value 10 for description.

T-0-0057 Kinematic Value 8

See T-0-0059 Kinematic Value 10 for description.

T-0-0058 Kinematic Value 9

See T-0-0059 Kinematic Value 10 for description.

T-0-0059 Kinematic Value 10

Default: based on kinematic

Attributes: Floating point, Read/Write in Parameter Mode.

These parameters are coefficients used in the kinematic equations of coordinated motion. Kinematics are unique to each application.

Coordinated Motion Status (0100-0113)

These read-only parameters provide status values for each task.

T-0-0100 Target Point Number

Attributes: Integer, Read-only

This parameter shows the current target point. For example, if 10 is displayed here, motion to point ABS[10] or REL[10] is taking place, or the machine is currently at this point.

T-0-0101 Segment Status

Attributes: Integer, Read-only

Limit: 0-8

This parameter shows the status of the current segment. The segment status

codes are shown in the table below. The codes are valid for the current segment, excepting codes 0, 1 and 7 which are transitional or do not apply to the current segment. Use code 8 to check if the target position has been reached. Use code 6 to check if motion is halted due to a path/stop.

Code	Segment Status	Activity
0	Segment ready	Segment in queue for path planner
1	Acceleration	Acceleration in progress
2	Slew (constant speed)	Slew in progress (at target velocity)
3	Blending	Blending in progress
4	Target deceleration	Deceleration to target position in progress
5	Controlled stop	Controlled stop taking place (error, jog or path/stop)
6	Stopped	Motion has stopped (error, jog or path/stop)
7	At Target	Segment at target position
8	Done	Motion on the segment is complete

T-0-0102 Rate Limit Status

Attributes: 8-bit Integer, Read-only

Limit: 0-7

Codes: 0 = No rate limiting

bit 0 =1 → velocity rate limiting on → value displayed = 1

bit 1=1 → acceleration rate limiting on → value displayed = 2

bit 2=1 → deceleration rate limiting on → value displayed = 4

Note: More than one rate limiting status can be active at once.

This parameter indicates if the coordinated motion path planner is performing rate limiting. If any of these bits are set to "1," the CLC is limiting the motion profiles of axes in the kinematic to maintain the constraints set by the axis maximum velocity, acceleration and deceleration parameters (A-x-0020, A-x-0021, A-x-0022). The limitations are the result of a command based on the task maximum velocity, acceleration and deceleration, and the point speed, acceleration and deceleration being higher than the axis maximum.

T-0-0111 Current X Position

Units: World Position

Motion Type: Coordinated

Attributes: Floating point, Read-only

This parameter returns the current commanded position of the X axis in world coordinates.

T-0-0112 Current Y Position

Units: World Position

Motion Type: Coordinated

Attributes: Floating point, Read-only

This parameter returns the current commanded position of the Y axis in world coordinates.

T-0-0113 Current Z Position

Units: World Position

Motion Type: Coordinated

Attributes: Floating point, Read-only

This parameter returns the current commanded position of the Z axis in world coordinates.

Task Status (0120-0200)

T-0-0120 Task Operating Mode

Codes: 0= initialization

1= parameter

2= manual

3= automatic

Attributes: Integer, Read-only

This parameter returns the current operating mode of the task as a code number. Information about the task's state, etc., is available in the CLC's Control and Status I/O registers. ***See Chapter 2. CLC I/O Systems for further information on the Control and Status I/O registers***.

T-0-0122 Task Diagnostic Message

Codes: 200-399: Warning error

400-599: Shutdown error

Attributes: ASCII string, Read-only

This parameter returns the current diagnostic message and/or code. During normal operation, a Message/Diag statement in the user program sets this message. If an error occurs during task execution, this diagnostic message is overwritten with an error message.

Examples:

```
Picking up the part

|_ Message for Operator from message/diag

421 User Program Stack Overflow

|   |_ Task Error Message

|_ Error code generated by CLC
```

T-0-0123 Task Status Message

Attributes: ASCII string, Read-only

This parameter returns the current status message for this task. A Message/Status command in the user program sets this message as an aid to the operator or for debugging purposes. This message is not overwritten with an error message, allowing debugging of an error condition set in the Task Diagnostic Message.

Examples:

```
Executing Gripper Routine

|_ Debugging Message from message/status

No active message

|_ Default when no message is set in the program
```

T-0-0130 Current Instruction Pointer

Attributes: Hex Long, Read-only

This parameter returns a hexadecimal value equal to the current task's execution address (i.e. the instruction pointer). The hex value is an offset from the start of the program. For example, "0x000000F0" indicates that the program counter is at 0xF0, or 240 bytes from the start of the program.

T-0-0131 Current Instruction

Attributes ASCII string, Read-only

This parameter returns the mnemonic for the current instruction and the first 2 arguments of the instruction. The mnemonic is in the base code format generated by the CLC compiler. This parameter is primarily used for debugging and troubleshooting programs.

Example:

```
003C AXIS_WAIT
```

T-0-0132 Instruction Pointer at Error

Attributes: Hex Long, Read-only

As a future enhancement, the CLC can possibly execute an error routine if an error in a task occurs. This status parameter returns the instruction pointer where the error occurred.

T-0-0133 Composite Instruction Pointer

Attribute: ASCII string, Read-only

This parameter returns a string of flags and pointers to determine program flow in the task. The first number is a flag denoting the instruction is of the task(1), a subroutine(2 to 11), or an event(-2 to -12).

For instructions in the main **task**, the next number is the instruction pointer.

Example:

>0 TP 1.133

>0 TP 1.133 1 009C $24

For instructions in a **subroutine** of the task, the flag contains the count of pointers. Except for the last number, all pointers are to subroutine calls, starting in the main task. The last number is the instruction pointer in the subroutine.

Example 1:

>0 TP 1.133

>0 TP 1.133 2 0034 009C $24

Main task subroutine call and pointer in subroutine.

Example 2:

>0 TP 1.133

>0 TP 1.133 4 0034 009C 1205 1344 $24

Main task subroutine call, second call, third call, and pointer in subroutine.

For instructions in an **event** of the task, the flag contains the count of pointers. Except for the last two numbers, all pointers are to subroutine calls, starting in the main task. The second last number is the instruction pointer when interrupted. The last number is the instruction pointer in the event.

Example 1:

>0 TP 1.133

>0 TP 1.133 -2 0034 009C $24

Main task instruction pointer and pointer in event.

Example 2:

>0 TP 1.133

>0 TP 1.133 -4 0034 009C 1205 1344 $24

Main task subroutine call, second call, pointer in subroutine, and pointer in event.

T-0-0135 Current Subroutine

Attribute: ASCII String, Read-only

This is a string which indicates the current subroutine being executed with the function number and name. This includes tasks, non-accessible

functions, functions, and subroutines. If function number and name information is not included in the user program file, the string “NONE” is printed.

To read data:

```
>0 TP x.135              ;Host asks for current subroutine for task ‘x’.
                         (x=A-D or 1-4)
>0 TP x.135 2 test_sub   ;The CLC responds with function number 2,
                          subroutine ‘test_sub’
```

T-0-0136 Stack Variable Data

Attribute: Data string, Read-only

This is a string of data of current stack variables data. Stack variables are valid only while the program flow is within a task or subroutine. Maximum number of stack variables is 16. If there are no arguments or local variables in a task or function, the string “NONE” is returned.

To read data:

```
>0 TP x.136              ;Host ask for stack variables for task ‘x’.
>0 TP x.136 1.22 5 7.998 1 0 $24
                         ;The CLC responds with values of all stack
variables.
```

To get the types of these variables, use the “SR” command in the serial protocol with the current function number from Tx.135. This command returns a string for each of a function argument’s attributes.

T-0-0137 Subroutine Breakpoint

Attribute: Integer, Read/Write

This task parameter specifies the index of the subroutine to halt program execution at when task breakpoint is enabled. Task breakpoint enable is bit 11 of the “Task_Control” register. Task program flow continues with a 0->1 transition on “Cycle_Start” bit of the “Task_Control” register. To get the count and names of subroutines of a program use:

```
>0 ST 0.0                ;Host ask for count of functions of currently active
                          program
>0 ST 0.0 23 $24         ;CLC return count of 23

>0 ST 0.1                ;Host ask for function at index 1.
>0 ST 0.1 Task_A $24     ;CLC return “Task_A”
```

The list of all function of a program can thus be uploaded. To set a breakpoint put the function index in the “Task Subroutine Breakpoint” parameter.

```
>0 TP x.137 5 $24        ;Host put function index 5 in task x breakpoint
```

parameter.

>0 TP x.137 $24

T-0-0138 Sequencer Information

Attribute: String, Read Only

This task parameter displays information about the currently running sequencer in both index and name format. The index indicates the current row in the sequence list or step list. If no sequencer is running, the string “NONE” is printed.

Otherwise, the CLC responds in the following format:

sequence_number sequence_row step_row sequence_name step_name function_name

Example:

>0 TP x.138 ;Host asks for sequencer information

>0 TP x.138 1 3 2 sequence_one Open_Mold move_fwd

;CLC responds with row numbers and names

T-0-0200 Last Active Event Number

Attributes: Integer, Read-only

This parameter returns the index of the current or last active event in the event (EVT) table. The value can be used to access other information (message, status, function, etc.) contained in the event table.

For example, if "99" is returned, EVT[99] is either executing or was the last event completed for this task.

Task Parameter Lists (2000-2001)

T-0-2000 List of All Parameters

This is a multiple-step List Parameter that requests a list of all the parameter numbers in this parameter set. The 'D' subclass of the serial port protocol is used to list these parameters.

T-0-2001 List of Required Parameters

This is a multiple-step List Parameter that requests a list of all the parameters stored in nonvolatile CLC RAM that must be set for proper CLC operation. The 'D' subclass of the serial port protocol is used to list these parameters.

3.6 Axis Parameters

Class A

The CLC Axis Parameters are used to configure the axis and provide limits for coordinated motion. Some parameters only apply to a specific axis mode (coordinated or single-axis). This is noted by the Motion Type in each parameter description.

Axis Setup (0001-0038)

A-0-0001 Task Assignment

Type:	Integer
Codes:	0 = No task selected
	1 = Task A
	2 = Task B
	3 = Task C
	4 = Task D
Motion Type:	Coordinated and Single Axis
Default:	0, or axis selected in program
Attributes:	Write in Parameter mode only
	(Automatically set when program activated.)

This parameter associates an axis with a user task. The TASK/AXES command and the Axis Setup Icon automatically set the parameter when a program is activated.

A-0-0002 Type of Positioning

Codes:	0= Normal Positioning
	1= Lagless Positioning
Default:	1 (lagless)
Attributes:	Integer, Write in Parameter Mode Only

This selects the SERCOS drive control mode. For coordinated, ELS phase, Cam, and single-axis motion, it switches between normal and lagless positioning modes.

A-0-0003 Axis Motion Type

Selections: 0 = Disabled
1 = Single Axis
2 = Coordinated Axis
3 = Velocity Mode
4 = Ratio Slave
5 = ELS Slave
6 = Torque Mode
7 = CLC Cam Axis

Default: 0, or axis selected in program

Attributes: Integer, Write only when the CLC is in Parameter Mode
(Automatically set when the user program is compiled)

This parameter selects the type of motion for an axis and enables the axis/drive address on the SERCOS ring. The applicable SERCOS parameters are automatically sent to the drive. If a user program is present on the card, this parameter is automatically set at exit from parameter mode based on the commands that set up coordinated or single-axis motion.

Single Axis runs the axis as a single, non-coordinated axis. The CLC sends only the position to the intelligent DDS drive. The DDS drive generates and controls the position motion profile.

Coordinated Axis is used for coordinated circular or linear motion profiles generated by the CLC path planner. The axes used for each path are determined by user program statements.

Velocity Mode operates the axis at constant velocity without a position control loop. The CLC sends only velocity commands to the DDS drive. The intelligent DDS maintains the velocity profile.

Ratio Slave designates the axis as a slave to the master axis designated by Parameter A-0-0030. The axis' drive operates in position loop mode, with its commanded velocity equal to the product of the master axis velocity times a ratio. The drive remains in position loop mode and at the commanded velocity at all times, even when the user task is not running.

ELS Slave designates the axis as an Electronic Line Shaft (ELS) slave.

Torque Mode Axis: The axis runs in torque modes, with torque commands sent from the CLC and no velocity or position loop.

CLC Cam Axis: The axis runs from a cam table stored on the CLC and uses the ELS virtual or real master.

A-0-0004 Axis Options

Defaults: 0000000000000001 (for single-axis, velocity, and coordinated axes)
0000000000000011 (for ELS axes)

Motion Type: All types

Attributes: Binary Word, Non-volatile, Write in Parameter Mode only or at compile time

Bits 1-16:

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Bit 1: Position Initialization

0 = Reset feedback in Phase 2

1 = Keep feedback value in Phase 2

Default: (1)

When a single-turn absolute or incremental encoder is used with the Indramat DDS drive, the drive defaults to resetting the encoder to a one-turn measurement each time it exits SERCOS Phase 2. The DDS includes a parameter "Starting Position Value" to override the default and initialize the feedback to the value specified.

When the Position Initialization bit is set to 1, the CLC sets the drive's Starting Position Value to the Position Feedback, and the drive's position remains the same after a switch out of parameter mode. When the Position Initialization bit is set to 0, a different Starting Position Value can be entered through the SERCOS service channel, or the feedback value will be reset to the default one-revolution measurement.

This parameter does not affect a multi-turn absolute encoder feedback. To set the multi-turn encoder's position, follow the procedure described in the DDS manual, using the "Set Absolute Position Measurement" and homing commands.

Bit 2: Positioning Mode

0 = Linear Positioning Mode

1 = Rotary Positioning Mode

Default: (0)

The positioning mode is valid for all axis types. All relevant scaling parameters are automatically set in the drives. If an axis is an ELS slave, its mode is automatically set to rotary positioning.

Linear Positioning Mode

When this bit is set to 0, linear positioning is selected. The units and scaling are specified with the Axis Position Units parameter. Absolute positioning is enabled in the drive.

Rotary Positioning Mode

When this bit is set to 1, rotary positioning is enabled. The position, velocity, and acceleration units and scaling are fixed at the drive. Modulo positioning is enabled by default, with a rollover value specified in Drive parameter S-0-0103.

When rotary positioning mode is enabled in the drive, position is in degrees, velocity in RPM, and acceleration in radians/sec. Single-axis and velocity mode values are entered in these units. Events for single-axis and velocity mode work with both rotary and linear positioning.

Coordinated motion and events are compatible with rotary positioning only if motion takes place within the modulo value and does not roll over. If an axis is coordinated, velocity is in linear units/sec and acceleration in units/sec.

Bits 3 and 4: ELS Synchronization Mode

This option sets the type of synchronization for an electronic line shaft axis. These bits are set automatically at program activation by the ELS/INIT command.

Bit 4	Bit 3	Mode
0	0	Velocity Synchronization The axis runs in velocity mode, with its velocity equal to the master velocity times the translation ratio. The ratio can be fine-adjusted at run time.
0	1	Phase Synchronization The axis runs in the operating mode selected in parameter As.2 (lagless positioning or positioning with lag), and maintains a phase relationship according to the phase offset and translation ratio parameters. The phase offset can be adjusted at run time.
1	1	Cam Table The axis is linked to a cam and synchronized to a master. Position relationship is maintained according to cam table, ratio and stretch factors. Position offset can be adjusted at run time.

Bit 5: *Reserved for future use*

Bit 6: Optional Cyclic Data

0= Use smallest cyclic data configuration

1= Include optional cyclic data (velocity feedback, programmed acceleration)

Default: (0)

The drives are set up for the smallest amount of cyclic data to increase the number of drives on the ring and increase cycle time. Velocity feedback is normally read through the service channel, which can take up to 50ms using a program command. To include velocity feedback in the cyclic data for all axis modes, set this bit to 1. In single-axis mode, programmed acceleration is also sent cyclically instead of through the service channel.

Up to five optional IDN's can be specified with parameters 180-196 in addition to or exclusive of this option bit.

Bit 7: Velocity Mode Ramp

0 = Velocity mode step command

1 = Velocity mode uses programmed acceleration

Default: (0)

When velocity mode is selected as an axis type, the drive commands the velocity that is sent to the CLC by default (bit=0). If bit 7 is set to 1, the CLC will accelerate and decelerate the axis as if it were in single-axis mode.

Bit 8: Reserved for Future Use

Bit 9: ELS Secondary Mode

0 = secondary mode is single-axis mode

1 = secondary mode is velocity mode

default: (0)

For an ELS axis, this bit selects the secondary, non-synchronized mode for a DDS drive. This is the default mode until the ELS/MODE command switches the axis into ELS velocity or phase synchronous mode.

When this bit is set to (0), the default secondary mode is single-axis positioning mode. Velocity mode may also be switched to using the ELS/MODE command, if it is configured in the cyclic data using parameters A-0-0180 Optional Command ID #1 to A-0-0196.

When it is set to (1), the secondary mode is velocity mode. On drives that don't support single-axis mode, the secondary mode is always velocity mode, regardless of this bit.

Bit 10: Disable ELS Shortest Path

0 = Shortest path positioning for ELS phase adjust

1 = Shortest path disabled

default: (0)

This bit selects the positioning method used for the ELS phase adjust move. When it is set to (0), the axis takes the shortest path within a revolution. If the difference in position is between 180 and 360 degrees, the axis travels counterclockwise.

When this bit is set to (1), shortest path is disabled. A move within one

revolution travels in the positive direction if the programmed position is positive, and in the negative if it is negative.

Bit 11: Positioning Using Secondary Encoder

0 = Use primary feedback (encoder 1)

1 = Use secondary feedback (encoder 2)

Default: (0)

This bit configures the drive to use Encoder 2 to close the position loop and provide cyclic feedback from drive parameter S-0-0053. This option must be set if the CLC is used with DDS-2.1 linear motor firmware.

Bit 12: Drive Disable Method

0 = Stop axis immediately

1 = Coast to a stop

Default: (0)

This bit changes the response of the drive after a fatal error or when the disable bit in the axis control register is set. It configures the way the CLC sets the enable bits (14 and 15) in the SERCOS control word (D-1.00134). When it is set to (0) (default), the drive immediately commands zero velocity before disabling torque and applying the brake. This option should be used for coordinated motion or linear motion, where coasting can cause damage or injury.

When it is set to (1), the drive immediately disables torque, causing the drive to coast, stopping with its own inertia. This should be used in some types of line shafting applications, where immediate disabling of the drive could cause damage.

Bit 13: Linear Axis Modulo Positioning

0 = Linear axis uses absolute positioning

1 = Linear axis uses modulo positioning

Default: (0)

When it is set to (0), absolute motion is enabled, with signed positions and no modulo. This option should be used for coordinated motion and most absolute positioning applications.

When it is set to (1), the modulo value in S-0-0103 is used. When the axis position reaches the modulo value, it resets to 0. There are no negative position values. Unlike rotary mode, the scaling of position, velocity, and acceleration is linear (inches or mm). This option should be used for continuous indexing operations.

Bit 14: Configure Minimum Cyclic Data

0 = Default or maximum cyclic data

1 = Minimum cyclic data

Default: (0)

ELS or Cam Axes

When set to (0), the velocity command is included in the cyclic data, which allows real-time update of velocity when the axis is in its secondary mode.

When set to (1), the velocity command is removed from the cyclic data. Because the velocity command is through the service channel only, no ramping or real-time control can be done on the axis. Therefore, the axis should only be run in ELS synchronization mode.

Single-axis mode

When set to (0), the programmed velocity is included in the cyclic data for applications where the velocity is changed often in the user program.

When set to (1), the programmed velocity is removed from the cyclic telegram to allow the maximum number of drives and options and the minimum SERCOS cycle time. The velocity is then sent through the service channel. This option is not recommended in applications where the velocity is changed often in the user program.

Bit 15: Disable Modulo Positioning for Rotary Axis

0 = Modulo positioning

1 = Absolute positioning

Default: (0)

When this bit is (1) modulo positioning is disabled if an axis is rotary. Positions less than zero are negative, and the modulo parameter is not used. This should be used only with positioning applications, not with indexing, ELS, or continuous motion. If the bit is set to (0) (default), the rotary axis position resets to the modulo value every revolution.

Bit 16: Disable Automatic Scaling

0 = CLC sets SERCOS scaling parameters (RECOMMENDED)

1 = CLC does not set SERCOS scaling parameters

Default: (0)

For most drives, the CLC sets SERCOS scaling parameters such as S-0-0044 and S-0-0076. If a drive does not accept the CLC's parameter settings, these parameters can be set manually. Note: This bit should only be set at the recommendation of the Indramat applications department.

A-0-0005 Linear Position Units

Selections:	0= inches
	1= millimeters
	2= radians
Default:	set automatically if coordinated motion, else 0 (inches)
Attributes:	Integer, Write in Parameter Mode only

This parameter selects the display and scaling units for linear axis position, speed, and acceleration data. No unit conversions are performed when changing this parameter.

The drive's display and scaling parameters are automatically set by the CLC. All data is sent in floating point format with the same resolution as the data sent to the drive. Velocity, acceleration, and jerk data is always sent with

three decimal place accuracy. The following decimal places are used for position data:

Units	Decimals	Max Value
inch	5	21474.83648
mm	4	214748.3648
rad	6	2147.483648

If an axis is in rotary mode, the drive automatically sets the position to degrees, the velocity to RPM, and the acceleration to radians/second/second. In rotary mode, the scaling and display units in this parameter do not apply.

If an axis is selected for coordinated motion, this parameter is automatically set on exiting parameter mode. If kinematic 1 is selected, it is set to the value stored in the Task Position Units parameter (Coordinated Motion (0005-0026)

T-0-0005 World Position Units). If any other kinematic is selected, it is set to radians (2) for use as a kinematic joint.

NOTE: *Changing this parameter affects the scaling of drive parameters (i.e., the feed constant). If you need to retain such constants, you must explicitly save each required drive mechanical parameter before changing the Linear Position Units parameter.*

A-0-0006 Reference Options

Attributes: Write in parameter mode only, Binary

Default: all bits = 0

This parameter selects options for reference position monitoring and homing.

Bit 1: Issue error when drive is enabled

When this bit is set to (1), the CLC immediately issues the error “500 Axis D is not referenced” if the drive is enabled while the position is not referenced. The CLC reads drive parameter S-0-0403, Position Feedback Status to determine if the position is referenced. While the drive is disabled, the Set Absolute Encoder procedure can be used to set the reference position.

This option should be set only when an absolute encoder is used, since it will prevent the incremental homing procedure from being used.

Bits 2-16: Reserved for Future Use

Other options will be added in the future.

A-0-0007 Configuration Mode

Default: 0

Attributes: Integer, Write in Parameter Mode only

This parameter allows drives to be excluded from the user program and initialized to single-axis or velocity mode. An error will not be issued if the drive is not found on the SERCOS ring. When this parameter is set to (0), the axis can be used in the program with its defined axis type, and its presence on the ring will be verified. When it is set to (1), the drive is excluded from the program, but can be jogged using default values. When it is set to (2), the drive is not configured, and is put into a torque-free mode after initialization. The default value of this parameter is (0). It can be written in parameter mode only.

A-0-0007	Selection in user program	Description
0	0	Drive can jog in default mode
1	0	Drive can jog in default mode
2	0	Drive not configured, motion cannot be performed
0	nonzero	Drive is configured according to selection in A-0-0003, presence on ring is checked.
1	nonzero	user program selection is bypassed, drive can jog in default mode
2	nonzero	user program selection is bypassed, motion cannot be performed

A-0-0009 Drive PLS Register

Default: 0 (disabled)

Motion Type: All types

Attributes: Integer, non-volatile, Read/write at any time

This parameter selects the register to which the drive-based PLS will be associated. The register selected here can be read by the user program, the I-O mapper, or the user interface as a status of the current PLS outputs. If this parameter is set to 0, the drive-based PLS for this axis is disabled.

If the register is associated with a DEA card, the drive internally updates the outputs every 2ms. If the register is not assigned to a DEA, this register will updated with the PLS status bits read from the drive every SERCOS cycle.

SERCOS config. note: The PLS Status IDN P-0-0135 will be placed in the AT in the cyclic data, with 2 bytes allocated. Parameter P-0-0124 (Assign IDN->DEA) is automatically set by the CLC if this register is a DEA output register.

A-0-0010 Start of Drive I-O

Default: for Axes 1-8, automatically set to defaults (see I-O description in manual)
for Axes 9-40, set to 0 (no register allocated)

Motion Type: All types

Attributes: Integer, non-volatile, Read/write in Parameter Mode only

This parameter selects the starting register number that correspond to the I-O cards that reside in the drive. The configuration for an Indramat DIAX02 or DIAX03 drive is shown below:

Register Number	Parameter used to Enable	Description	I-O Card
A-0-0010	A-0-0011	Inputs, Card 1	DEA4.1
(A-0-0010) + 1	A-0-0011	Outputs, Card 1	DEA4.1
(A-0-0010)+ 2	A-0-0014	Inputs, Card 2	DEA5.1
(A-0-0010)+ 3	A-0-0014	Outputs, Card 2	DEA5.1
(A-0-0010)+ 4	A-0-0017	Inputs, Card 3 or Analog Input 1	DEA6.1 or DRF
(A-0-0010)+ 5	A-0-0017	Outputs, Card 3 or Analog Input 2	DEA6.1 or DRF

A-0-0011 Drive I/O Card 1 Setup

Type: Integer

Codes: 0= disabled
1= enable DEA 4.1 parallel I-O

Default: 0

Attributes: Read/Write in Parameter Mode Only

The Indramat DDS digital drive can include up to three optional parallel I/O cards that provide 16 outputs and 15 inputs. If this parameter is set to '1', the first I/O card is enabled and its contents are transmitted in the cyclic telegram to and from the corresponding I/O registers.

A-0-0014 Drive I/O Card 2 Setup

Type: Integer

Codes: 0= disabled
1= enable DEA 5.1 parallel I-O

Default: 0

Attributes: Read/Write in Parameter Mode Only

The Indramat DDS digital drive can include up to three optional parallel I/O cards that provide 16 outputs and 15 inputs. If this parameter is set to '1', the second I/O card is enabled and its contents are transmitted in the cyclic telegram to and from the corresponding I/O registers.

A-0-0017 Drive I/O Card 3 Setup

Type:	Integer
Codes:	0= disabled
	1= enable DEA 6.1 parallel I-O
	2= enable analog input 1 on DRF card
	3= enable both analog inputs on DRF
Default:	0
Attributes:	Read/Write in Parameter Mode Only

This parameter is used to select DDS analog or parallel inputs. Options allow selection of a DEA6.1 parallel I-O card (1), analog input 1 on the DRF card (2), or analog inputs 1 and 2 (3). Cyclic data can be optimized if only one analog input is needed by selecting (2) instead of (3).

Analog Inputs:

The analog inputs on the DRF card are 16-bit values which can be read as parameters or from I-O registers. The inputs are converted and scaled by the DDS according to the amplification selected on the connector. The analog input 1 value is stored in the register for parallel input 3 and the analog input 2 in the parallel output 3 register. For example, on the CLC 1.1, if A-0-0017 Drive I/O Card 3 Setup is set to 2, analog input 1 value is in register 44. If parameter A2.17 is set to 3, analog input 1 value is in register 50, and analog input 2 value in register 51.

A-0-0020 Maximum Velocity

Units:	Axis Velocity (units/min)
Default:	1000.0
Motion Type:	All types
Attributes:	Floating point, Read/write at any time

The axis velocity is limited to this value during a coordinated move. Note that the Bipolar Velocity limit on the drive (S-0-0091) must also be set, and it is independent of this parameter. For single-axis or velocity mode motion, this is the maximum velocity that can be programmed.

A-0-0021 Maximum Acceleration

Units:	Axis Acceleration (units/sec^2)
Default:	200.0
Motion Type:	All types
Attributes:	Floating point, Read/write at any time

The axis acceleration is limited to this value during a coordinated move. On the drive level, the acceleration capability is limited by the amount of peak torque that the drive and motor are rated for, and the Bipolar Acceleration limit parameter if it is set. For single-axis or velocity mode motion, this is the maximum acceleration that can be programmed.

A-0-0022 Maximum Deceleration

Units: Axis Acceleration (units/sec^2)
Default: 200.0
Motion Type: All types
Attributes: Floating point, Read/write at any time

The deceleration for this axis is limited to this value during a coordinated move. At the DDS drive level, the acceleration capability is limited to the rated drive or motor peak torque; and, if set, the drives' Bipolar Acceleration limit parameter.

A-0-0023 Jog Acceleration

Type: Integer
Units: Percent of Maximum Acceleration
Limits: 1% to 100%
Default: 100%
Attributes: Read/write at any time

This parameter sets the acceleration and deceleration rate for axis jogging in manual mode. It is entered as a percentage of the maximum acceleration (A-0-0021 Maximum Acceleration).

A-0-0025 Maximum Jog Increment

Units: linear: position units
rotary: revolutions per minute
Default: 1.0
Attributes: Floating point, Read/write at any time

This parameter defines the maximum distance used for incremental single-axis jogging. The Large Increment and Small Increment percent parameters are based on this value.

A-0-0026 Maximum Jog Velocity

Units: linear: position units per minute
rotary: revolutions per minute
Default: 100.0
Attributes: Floating point, Read/write at any time

This parameter defines the maximum velocity used for jogging an axis in single-axis mode, velocity mode, or as a joint. The system fast and slow axis percent parameters are multiplied by this value to calculate jog velocity.

A-0-0030 Ratio Mode Master Axis

Selections: 0 to maximum number of axes

Motion Type: Ratio Slave

Attributes: Integer, Write in Parameter Mode only

(Automatically set when the user program is compiled.)

This parameter displays the axis that is used as the master for ratio, ELS, or Cam functions. If an axis is slaved to the ELS virtual master, it is set to 0. If a real master is used, it is set to the axis on which the encoder resides. For the ratio function, the master axis is set independently for each slave at program compile time. The velocity feedback of the master is multiplied by the ratio established by parameters A-0-0031 CLC Cam/Ratio Master Factor (N) and A-0-0032 CLC Cam/Ratio Slave Factor (M).

A-0-0031 CLC Cam/Ratio Master Factor (N)

Limits: -100000 to 100000

Motion Type: Ratio Slave or Cam

Attributes: Floating point, Read/write at any time

(This parameter is also set by the AXIS/RATIO instruction.)

The slave-to-master ratio is defined using two parameters, A-0-0031 and A-0-0032 CLC Cam/Ratio Slave Factor (M). This allows the CLC to normalize the ratio calculation, preserving full system accuracy for repeating-decimal ratios such as 2/3. Both parameters must be set or a run-time error may occur.

In Ratio mode, the velocity of the slave is determined by:

$$V_{slave} = V_{master} * (K_{slave} / K_{master})$$

where:

V_{slave} = Velocity of the slave axis

V_{master} = of the master axis

K_{slave} = Slave factor set by Parameter A-0-0032

K_{master} = Master factor set by Parameter A-0-0031

A-0-0032 CLC Cam/Ratio Slave Factor (M)

Same format as parameter A-0-0031 CLC Cam/Ratio Master Factor (N)

A-0-0033 CLC Cam Stretch Factor (H)

Default: 1.0

Limits: -100000.0 to 100000.0

Motion Type: CLC Cam

Attributes: Float, Read/write at any time.

This is the stretch factor (H) for Cam motion on the CLC. Every position at the output of the cam is multiplied by this value.

A-0-0034 CLC Cam Currently Active

Default: 0

Limits: 0- max. number of cams
Motion Type: CLC Cam
Attributes: Integer, Read/write at any time

This is the cam number that is currently active for this axis. If the cam is set to 0, the axis directly follows the master axis. The cam activation only takes affect after the master has passed zero degrees or when the master is stopped.

A-0-0035 CLC Cam Position Constant (L)

Default: 0.0
Limits: -100000.0 to 100000.0
Motion Type: CLC Cam
Attributes: Float, Read/write at any time.

This factor for Cam motion on the CLC allows the master position to be added to the cam position, according to the equation:

Øs = L * Øm + H * CamTable(z)

A-0-0036 Ratio Mode Encoder Type

Attributes: Read/write in parameter mode only, integer
Units: None
Selections: 0 if not used or primary feedback is used
1 if primary feedback is used
2 if secondary feedback is used.

This option selects the type of master used for the CLC Ratio Mode. The primary encoder or secondary encoder can be used. Options (0) and (1) select the master drive's primary feedback, which is the value read from drive parameter S-0-0051. Option (2) selects the secondary feedback, read cyclically from drive parameter S-0-0053 (feedback 2). This parameter is set automatically from the axis/initialize instruction.

A-0-0037 Ratio Mode Step Rate

Format: Floating Point
Storage: volatile, read/write at any time
Motion Types: Ratio Mode
Default: 0.0 (instantaneous change)
Units: units/ second

This parameter sets the rate used in CLC Ratio Mode when the Ratio parameters A-0-0032 CLC Cam/Ratio Slave Factor (M) and A-0-0033 CLC Cam Stretch Factor (H) are changed, either directly or through the axis/ratio command. If the step rate is set to 0, the ratio will be changed immediately without a ramp.

Example: The current ratio is 0, the programmed ratio is 10:1, and the step rate parameter is set to 10 units/sec. The ratio will be ramped for one second until it reaches the target value.

A-0-0038 Ratio Mode Options

Attributes: Read/write in parameter mode only, binary word

Units: None

Default: All bits = 0

Note: Rightmost bit is Bit 1

The CLC card allows a Ratio Mode slave axis to run in velocity or position mode and follow a master axis in velocity or position mode.

Bit 1: Control Mode

This option selects the control loop used for the CLC Ratio Mode. When set to (0), the axis is in position mode while it is following the master. When set to (1), it is in velocity mode.

Position Mode (Bit 1=0): When an axis is in position mode, it will maintain the position of the master relative to when the slave was switched into ratio mode. A constant phase difference is maintained between the master and the slave. The drive is configured in position loop mode, with lag or lagless control selected in parameter A-0-0002.

Velocity Mode (Bit 1=1): When an axis is in velocity mode, a commanded velocity is maintained. The position offset may not be maintained, since no position loop is closed. This option is useful for applications where the velocity of the slave must be adjusted. The slave velocity may be adjusted in response to a tension loop by changing drive parameter S-0-0037, additive velocity command.

Bit 2: Feedback Mode

When this bit is set to (0 - default) the CLC will follow the difference in position feedback values from the master axis. When this bit is set to (1) the velocity feedback of the master axis will be followed.

Follow Position (Bit 2=0): This option should be selected when it is critical to maintain a relative position between the master and the slave. Position following is more accurate than velocity following due to conversion errors and drive implementations.

Follow Velocity Bit (2=1): This option should be selected when there are problems with position initialization when switching into ratio mode, and when relative accuracy of position is not required

Axis Status (0100-0145)

These read-only parameters provide status values for the specified axis.

Note: The feedback values are obtained from the drive through the cyclic SERCOS telegram rather than through the service channel. They should be read from the user program or user interface, since they will be updated much faster than using the service channel.

A-0-0100 Target Position

Units: Axis Position
Motion Type: Single-axis
Attributes: Floating point, Read/write at any time

DIAX02 Drives

This is the programmed position used by the drive in Single-axis mode, equivalent to parameter P-0-0049

DIAX03, ECODRIVE, SERCOS Standard

This is the programmed position used by the drive in Single-axis mode, equivalent to parameter S-0-0258.

A-0-0101 Commanded Position

Units: Axis Position
Motion Type: Coordinated
Attributes: Floating point, Read-only

This is the commanded position used for coordinated moves, the cyclic equivalent of SERCOS parameter S-0-0047.

A-0-0102 Feedback Position

Units: Axis Position
Motion Type: All types
Attributes: Floating point, Read-only

This is the position feedback from the drive, equivalent to SERCOS parameter S-0-0051 in Phase 4. It is not updated in phases 0-3.

A-0-0110 Programmed Velocity

Units: Axis Velocity (units/min)
Motion Type: All types except coordinated
Attributes: Floating point, Read/write at any time

DIAX02 Drives

This is the programmed velocity used by the drive in Single-axis mode, equivalent to parameter S-0-0091.

DIAX03, ECODRIVE, SERCOS Standard

This is the programmed velocity used by the drive in Single-axis mode, equivalent to parameter S-0-0259.

A-0-0111 Commanded Velocity

Units:	Axis Velocity (units/min)
Motion Type:	Coordinated and Velocity
Attributes:	Floating point, Read/write at any time (Automatically set when the user program is compiled.)

This is the commanded velocity, which the drive generates in coordinated mode, and the CLC generates in Velocity mode. It is equivalent to SERCOS parameter S-0-0036.

A-0-0112 Feedback Velocity

Units:	Axis Velocity (units/min)
Motion Type:	All types
Attributes:	Floating point, Read-only

This is the velocity feedback from the drive, equivalent to SERCOS parameter S-0-0040 in Phase 4.

NOTE: *If velocity isn't configured in the cyclic telegram, the CLC reads this value from the service channel.*

A-0-0120 Programmed Acceleration

Units:	Axis Acceleration (units/sec^2)
Motion Type:	All types except coordinated
Attributes:	Floating point, Read/write at any time

DIAX02 Drives

This is the programmed acceleration used by the drive in Single-axis mode, equivalent to parameter S-0-0138.

DIAX03, ECODRIVE, SERCOS Standard

This is the programmed acceleration used by the drive in Single-axis mode, equivalent to parameter S-0-0260.

A-0-0131 SERCOS Control Word

Motion Type: All

Attributes: Binary word, Read-only

This is the current value of the SERCOS master control word (S-0-0134).

A-0-0132 SERCOS Status Word

Motion Type: All

Attributes: Binary word, Read-only

This is the current value of the SERCOS drive status word (S-0-0135).

A-0-0133 AT Error Count

Attributes: Integer, Read-only

The AT error counter can be used for troubleshooting of SERCOS connections. If the value of this parameter is increasing while it is being displayed, there may be a noisy SERCOS connection or a faulty communication card on the drive associated with this axis.

If two consecutive AT's are invalid, the CLC will issue a SERCOS disconnect error. This counter counts up to a maximum of 65535. If a large value is set in the counter, there may have been a noisy transmission over a long period of time.

A-0-0140 Mfg. Class 3 Status Word

Type: Binary Word

Motion Type: Single-axis, ELS

Attributes: Read-only

This is the cyclic equivalent of parameter S-0-0182 on Indramat drives. It is always included in the cyclic data in single-axis mode. In ELS mode, it is included only if the phase adjust is executed on the drive. The CLC sets the axis status register based on some of these bits.

Bits (1=LS bit)

1	AS input is active (DIAX03, ECODRIVE)
5	Set if target position exceeds position limit value
6	Set if programmed velocity exceeds max motor speed
7	Set if axis is in-position and target position reached
9	Set if ELS phase adjust or velocity adjust is complete (DIAX03)
12	Set if drive halt was completed

A-0-0141 Torque Mode Commanded Torque

Units: Percent of motor standstill (stall) torque

Attributes: Floating point, Read/write at any time

This is the cyclic equivalent of drive parameter S-0-0080, Torque Command. It is set with a torque command from the CLC user program when an axis is in Torque Mode.

A-0-0142 Torque Feedback (cyclic)

Units: Percent of motor standstill (stall) torque

Attributes: Floating point, Read-only

This is the cyclic equivalent of drive parameter S-0-0084, Torque Feedback. It is updated only in Torque Mode.

A-0-0145 Current Motion Type

Codes: See Parameter A-0-0003 Axis Motion Type

Attributes: Integer, Read-only

The CLC switches ELS, Cam, Ratio, and Torque Mode axes to different operating modes for jogging and setup. This parameter displays the current motion type corresponding to the drive's operating mode. For an ELS axis in synchronization mode, Ratioed Axis (4) is printed. When it is not in sync. mode, it is either in single-axis (1) or velocity (3) mode.

Electronic Line Shaft (0150-0164)

A-0-0150 Programmed Ratio Adjust

Units: Percent

Default: read from drive at switch to phase 4

Limits: -100.0 (stopped) to 300.0 (4 times master speed)

Attributes: Floating point, read/write at any time

This is the ratio fine adjust parameter, corresponding to drive parameter P-0-0083. It is adjusted from -100 to 300 percent using the ELS/ADJUST command. If fine adult is included in the cyclic data, it is updated every SERCOS cycle and may be adjusted using a ramp (A-0-0159 Ratio Adjust Step Rate). Otherwise, the drive's service channel is used.

A-0-0151 Programmed Phase Offset

Units: World Position

Attributes: Floating point, Read/write at any time

This is the programmed phase offset value used in position synchronization

and drive cam modes. The drive executes a position profile with acceleration (P-0-0142) and velocity limit (P-0-0143) when this parameter is changed. This value corresponds to drive parameter S-0-0048, in the cyclic data. Set through the ELS_Adjust or Cam_Adjust command.

Note: *When dynamic synchronization is enabled on the drive, parameters A-0-0152 through A-0-0163 are not used.*

A-0-0153 CLC Phase Adjust Average Velocity

Units: RPM

Default: 10.0

Attributes: Floating point, Read/write at any time

When the phase offset is changed, the CLC performs an absolute positioning move in addition to the ELS master command. This is the target velocity used for the phase offset adjustment move.

The time constant parameter A-0-0155 CLC Phase Adjust Time Constant can be used to automatically set the velocity for the phase offset based on the time of the move. When A-0-0153 is used, it sets the average velocity of the phase offset move. Note that the peak velocity will be up to twice as large as this value.

A-0-0155 CLC Phase Adjust Time Constant

Units: seconds

Default: 0.0 (disabled)

Limits: 0 to 60

Attributes: Floating point, read/write at any time

The CLC uses a filter to implement a jerk limited profile for the phase adjust move.

This parameter sets the amount of time that the move will require, regardless of the position. This can be calculated in the user program so that the phase adjust is always distributed for the length of one part, for example.

If parameter A-0-0155 is set to 0, the CLC uses parameter A-0-0153 CLC Phase Adjust Average Velocity, Phase Adjust Velocity, to set the average velocity of the phase offset move. Note that the peak velocity will be up to twice as large as this value.

A-0-0157 Current Phase/ CLC Cam Master Offset

Units: World Position

Attributes: Floating point, Read-only

This parameter shows the current cyclic phase offset command sent from the CLC to the drive.

A-0-0159 Ratio Adjust Step Rate

Units: percent per second

Default: 0.0 (instantaneous change)

Limits: 0.0 to 10000.0

Attributes: Floating point, read/write at any time

This parameter sets the rate used when the ELS Programmed Ratio Adjust, A-0-0150 Programmed Ratio Adjust, is changed.

If the step rate is set to 0, the ratio adjust will be changed immediately without a ramp. If ratio adjust is not included in the cyclic data for the drive, the value is not ramped. See A-0-0004 Axis Options and A-0-0180 Optional Command ID #1 for data configuration.

Example: The current ratio adjust is 0%, the programmed adjust is 10%, and the step rate parameter is set to 10 %/sec. The ratio adjust will be ramped for one second until it reaches the target value.

A-0-0160 Commanded Ratio Adjust

Units: Percent

Attributes: Floating point, read only

This is a status of the currently commanded value of ratio adjust, which is updated either gradually or immediately to the Programmed Ratio Adjust value, A-0-0150 Programmed Ratio Adjust.

A-0-0161 CLC Cam Programmed Slave Adjust

Attributes: Read/write, floating point

This is the target value for the slave phase adjust (**Sph** in cam equation), which is set using the CAM/ADJUST instruction.

A-0-0162 CLC Cam Current Slave Adjust (Sph)

Attributes: Read only, floating point

This is the currently commanded value of the slave phase adjust (**Sph** in cam equation).

A-0-0163 CLC Cam Output Position

Attributes: Read only, floating point

This is the slave position, which is the output of the cam equation for this axis. It provides a status of the cam position even when the axis is not synchronized to the cam. A initial move or phase offset can be performed with the value read from this parameter before switching into cam synchronization mode.

A-0-0164 ELS Options

Format: Binary Word

Storage: Non-volatile, Write in Parameter Mode only or at program activation

Motion Types: ELS and cam

Note: Bit 1 is the rightmost bit.

Default: 0000000000000001 (To enable default drive based offset)

This parameter sets several options for ELS and cam motion. For dynamic synchronization and ramp up - lock on bits 1 and 6 should be set to 1.

Bit #	Description
1	Use drive internal phase offset
2	Remove phase offset from cyclic data
3	Remove 'mfg class 3 diag' from cyclic data
4	Remove 'target position' from cyclic data
5	Use service channel for cam control/status
6	Do not automatically set phase offset when switching into synchronization

Bit 1: Use drive internal phase offset

(0) = Phase offset profile is generated by CLC

(1) = Phase offset profile is generated by the drive.

Default: (0)

On Indramat DIAX03 drives, the phase offset for ELS and drive-based cams can be performed on the drive.

Enabling the phase offset on the drive frees CLC resources.

Notes:

1. The CLC filter and other related parameters and bits are disabled with this option.
2. Enabling the phase offset on the drive automatically places parameter S-0-0182 into the cyclic data so that the 'phase adjusted' bit in the axis status register can be updated with bit 8 of this parameter.
3. When the phase offset is performed on the drive, the 'phase adjusted' status in the axis status register depends on parameter S-0-0228 (position synchronization window).

Bit 2: Remove phase offset from cyclic data

(0) = Phase offset is in cyclic data

(1) = Phase offset is through service channel

Default: (0)

If the phase offset is never changed while the user program is running, cycle time in the SERCOS ring can be conserved by eliminating it from the cyclic data. When the phase offset is sent through the service channel, the user program command will take up to 50ms to execute. If the CLC is generating the phase offset profile, the value will change instantly.

Bit 5: Use service channel for cam control/status

(0) = Cam control and status uses drive real time bit

(1) = Cam control and status is through service channel

Default: (0)

In most applications, this bit should be set to (0). The cam can then be changed at the following SERCOS cycle through the drive real time bits. The drive has two real time bits that are used for cams and probe functions. If the cam does not need to be changed on the fly and more than one probe is needed for registration, this parameter bit can be set to (1).

Bit 6: Do not automatically set phase offset

(0) = Set phase offset automatically at ELS synchronization

(1) = Phase offset can be initialized with programmed offset (A-0-0151 Programmed Phase Offset) before synchronization.

Default: (0)

If this bit is set to (1), the phase offset can be initialized to any value before the drive is switched into ELS mode. If the bit is set to (0), the CLC automatically establishes relative phase synchronization.

Axis Feedback Capture (Registration 0170-0174)

A-0-0170 Probe Configuration Status

Attributes: Binary Word, read only

This parameter shows the status of feedback capture set up by program activation or execution. Refer to the Feedback Capture Events in the Events section of the Programming Elements chapter. For a description of the SERCOS probe functions available with DDS-2 drives, refer to the DDS-2 manual.

Bits 1-4: Probes Enabled

Bit 1: 0 = probe 1 positive edge disabled

1 = probe 1 positive edge enabled from capture/setup command

Bit 2: 0 = probe 1 negative edge disabled

1 = probe 1 negative edge enabled from capture /setup command

Bit 3: 0 = probe 2 positive edge disabled
1 = probe 2 positive edge enabled from capture /setup command

Bit 4: 0 = probe 2 negative edge disabled
1 = probe 2 negative edge enabled from capture/setup command

A-0-0171 Probe 1 Positive Captured Value

Units: Axis position (linear), or degrees (rotary)
Attributes: Floating point, Read-only, cyclic data

The parameter shows the last captured value for each edge of the probe inputs on the DDS drive. Refer to Feedback Capture Events in the Events section of the Programming Elements chapter.

A-0-0172 Probe 1 Negative Captured Value

See the description for Parameter A-0-0171 Probe 1 Positive Captured Value.

A-0-0173 Probe 2 Positive Captured Value

See the description for Parameter A-0-0171 Probe 1 Positive Captured Value.

A-0-0174 Probe 2 Negative Captured Value

See the description for Parameter A-0-0171 Probe 1 Positive Captured Value.

Optional SERCOS Data (0180-0196)

A-0-0180 Optional Command ID #1

See A-0-0182 Optional Command ID #3

A-0-0181 Optional Command ID #2

See A-0-0182 Optional Command ID #3

A-0-0182 Optional Command ID #3

Default: 0 (disabled)
Attributes: Integer, read/write in parameter mode only

The CLC automatically configures SERCOS with cyclic data according to the axis type. In addition to the default cyclic data, up to two optional drive parameters can be added for update every SERCOS cycle.

These parameters select the IDN of the drive parameter to be transmitted cyclically. Any parameter supported by the drive can be written. Parameters

A-0-0190 and **A-0-0191**, and **A-0-0192 Command Data #3** are the real-time values corresponding to these parameters.

Example: To update the torque limit in real-time, set **A-0-0180** to 92 (Torque Limit). While the drive is in Phase 4, the value in Parameter **A-0-0190** is sent cyclically to the drive, and can be written using the parameter transfer instruction in the user program.

A-0-0185 Optional Feedback ID #1

See A-0-0186 Optional Feedback ID #2

A-0-0186 Optional Feedback ID #2

Default: 0 (disabled)

Attributes: Integer, read/write in parameter mode only

The CLC automatically configures SERCOS with cyclic data according to the axis type. In addition to the default cyclic data, up to two optional drive parameters can be added for update every SERCOS cycle.

These parameters select the IDN of the drive parameter to be received cyclically. Any parameter supported by the drive can be read. Parameters A-0-0195 Feedback Data #1 and A-0-0196 Feedback Data #2 are the real-time values corresponding to these parameters.

Example: By default, feedback velocity is received through the service channel. To obtain the feedback velocity in real-time, set A-0-0185 to 40 (Feedback Velocity). While the drive is in Phase 4, the value in Parameter A-0-0195 is updated, and can be read using the parameter transfer instruction in the user program.

A-0-0190 Command Data #1

See A-0-0192 Command Data #3.

A-0-0191 Command Data #2

See A-0-0192 Command Data #3.

A-0-0192 Command Data #3

Attributes: Depends on A-0-0180 through A-0-0182 Optional Command ID #3 respectively, read/write at any time

These real-time values correspond to the selected optional cyclic parameter, as described in parameters **A-0-0180**, **A-0-0181** and **A-0-0182**.

A-0-0195 Feedback Data #1

See A-0-0196 Feedback Data #2

A-0-0196 Feedback Data #2

Attributes: Depends on respectively A-0-0185 & A-0-0186, read only

These real-time values correspond to the selected optional cyclic parameter, as described in parameters A-0-0186 Optional Feedback ID #2.

Axis Parameter Lists (2000-2001)

A-0-2000 List of All Parameters

Type: Parameter List
Attributes: Read-only

This is a multiple-step List Parameter that requests a list of all the parameter numbers in this parameter set. The 'D' subclass of the serial port protocol is used to list these parameters.

A-0-2001 List of Required Parameters

Type: Parameter List
Attributes: Read-only

This is a multiple-step List Parameter that requests a list of all the parameters stored in nonvolatile CLC RAM that must be set for proper CLC operation. The 'D' subclass of the serial port protocol is used to list these parameters.

3.7 Drive Parameters

Class D, Sets 1-8

The CLC SERCOS master conforms to SERCOS Version V 01.02, as does the Indramat DDS and RAC drives. Any drive that conforms to this specification should run in the CLC's Coordinated motion and Velocity modes. The CLC single-axis mode uses the internal intelligence of the Indramat DDS 2.1 drive and is product-specific. Consequently, DIAX01 drives will not run in single-axis mode. DIAX01 drives include: RAC and KDA drives.

A SERCOS-compatible drive contains many parameters needed to configure a motion system. Several of these parameters are used only by the controller to configure the communications, timing, and scaling. The CLC automatically sets the SERCOS timing and scaling parameters when it switches into Communication Phase 2.

Most servo and position loop parameters are contained on the SERCOS-compatible digital drive. These parameters include: Feed Constant, Kv Factor, In-Position Window, Monitor Window and all homing parameters. Acceleration, Deceleration and Jerk are defined in the user program and limits for these are set in the drive. A list of most often changed drive parameters is included in this manual. For a completed description of the parameters, refer to the DDS-2.1 drive manual.

The SERCOS and DDS-2.1 manuals name parameters according to the following format:

P-1-0100
| | |_ Parameter number
| |_ Parameter set 1-8
|_ Parameter type

The CLC parameters are classified according to type or location. Translating the above parameter, as it is referred to in the SERCOS manual, to a CLC parameter:

SERCOS user parameter base	32768
SERCOS parameter set 1 * 4096	4096
SERCOS parameter number	100
	36964

CLC parameter D 1.36964

The communication protocol allows the SERCOS "P" to be specified, but some decoding may have to be performed to translate between the two formats if there is more than one parameter set.

Drive Status

The most useful SERCOS status parameter is the drive diagnostic message, SERCOS Parameter 95 (S-0-0095). When a drive error occurs, the DDS drive sends a descriptive text message of up to 80 characters showing the status or error condition for the drive. Other drive status parameters are better accessed as axis parameters or should be used only when more specific diagnostics are needed. A complete list of the DDS drive diagnostic messages is included in the DDS 2.1/3.1 or DIAX03 manuals.

Required DDS Setup

Indramat's Visual Motion program provides a utility (*select Drives under the setup menu*) for setting up many of the parameters needed for system start-up and operation. Many of these parameters are essential for drive operation. The number of necessary parameters depends on the application. If homing is used, many more parameters need to be entered. ***See Appendix D. Drive Parameter Editor.***

SERCOS drives can provide lists of which parameters are required in each SERCOS phase, and which ones are invalid. A command is also included in the DDS to load default tuning values based on the connected motor and feedback method.

Default settings for new parameters *for DIAX03 drives with ELS-04Vrs firmware*:

The drive will not run at all or will issue errors if the following parameters aren't set to the correct default values below. The drive does not set valid defaults at initial start-up. These parameters should be set in addition to the ones needed for ELS-03.

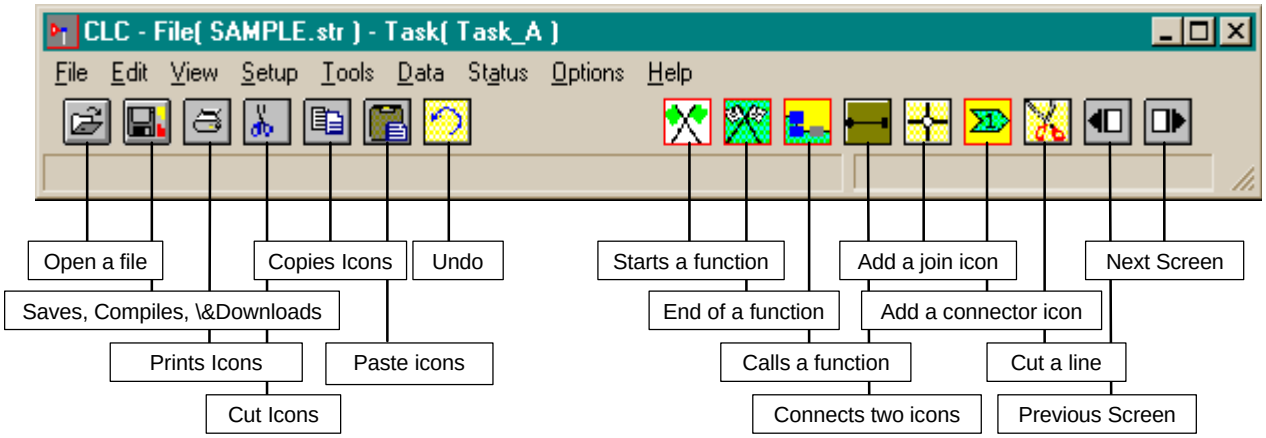
Parameter	Set to	Notes
P-0-0109 Torque/force peak limitation	S-0-0092	The drive sets this parameter to 0% by default, and the motor will not move at all.
S-0-0138 Bipolar acceleration limit	A-0-0021	This is the rate used for halting the drive. Its default value of 0 means that the drive will never stop.
P-0-0127 Overload warning	100%	This is initially set to 0 by the drive, and the warning E2-61 is continuously issued
S-0-0393 Modulo Format	0	If this is not initially set to 0 certain CAM functions will not work properly.

The following is a list of DDS parameters that typically need to be changed when configuring a CLC system. The list also includes ELS-04VRS parameters that have new identification numbers for compliance with the latest SERCOS specification. User interfaces and user programs should be changed if necessary to handle the new parameter numbers.

Parameter (ELS-03VRS)		ELS-04VRS	Description
S-0-0049	D 1.00049		Positive Position Limit Value
S-0-0050	D 1.00050		Negative Position Limit Value
S-0-0055	D 1.00055		Position Polarity Parameter
S-0-0057	D 1.00057		Position Window
S-0-0091	D 1.00091		Bipolar Velocity Limit Value
S-0-0092	D 1.00092		Bipolar Torque Limit Value
S-0-0100	D 1.00100		Velocity Loop Proportional Gain
S-0-0101	D 1.00101		Velocity Loop Integral Action Time
S-0-0104	D 1.00104		Position Loop KV-Factor (closed-loop control)
S-0-0121	D 1.00121		Input Revolutions of Load Gear
S-0-0122	D 1.00122		Output Revolutions of Load Gear
S-0-0123	D 1.00123		Feed Constant
S-0-0124	D 1.00124		Zero Velocity Window
S-0-0138	D 1.00138		Bipolar Acceleration
S-0-0159	D 1.00159		Monitoring Window
P-0-0004	D 1.32772		Smoothing Time Constant
P-0-0005	D 1.32773	S-0-0265	Language Selection
P-0-0019	D 1.32787		Starting Position Value
P-0-0020	D 1.32788	S-0-0298	Reference Cam Shifting
P-0-0050	D 1.32818	S-0-0348	Proportional-Gain Acceleration Feed Forward
P-0-0120	D 1.32888	S-0-0391	External Encoder Monitoring Window

4 VisualMotion Menu Commands

4.1 Introduction



This chapter is a reference for VisualMotion's menu commands:

The File Menu
The Edit Menu
The View Menu
The Setup Menu
The Tools Menu
The Data Menu
The Status Menu
The Options Menu
Help

Some of the frequently used menu commands also have short cut buttons represented on the screen. Some of the commands described in this chapter are discussed in further detail in *Chapter 5 - **Programming Concepts***.

4.2 The File Menu

Standard Windows file commands, compiling CLC program, archiving CLC program and variables, and printing CLC programs.

New - Opens a new icon or textual language program.

Open - Opens a existing icon or textual language program.

Save - Saves a named icon program to a file.

Save As - Saves a icon program to a file, filename is prompted for.

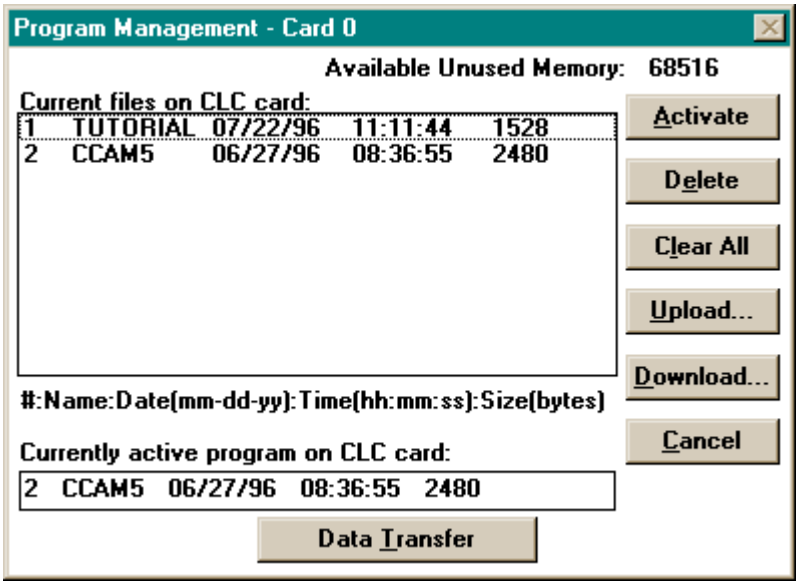
Save, Compile, \Download

Compile - Checks and converts icon or textual language program to code CLC can interpret.

Display Code - Displays the CLC "Base Code", intermediate code generated by the first pass compiler and used by second pass compiler. This is only for reference to errors generated by the second pass compiler.

Program Management

The CLC Program Management dialog box provides the facilities for downloading compiled programs to the CLC card, uploading executable programs for archiving on the Host systems hard drive, activating and deleting CLC resident programs, and transferring data sets between the active program and another resident program.



To download a program, simply click the **Download** button and select the desired file. Executable programs are stored with the ".exc" file extension together with other project files in the Project Directory, which is defined using the Setup/Configuration command . Once the file is selected, another dialog box is displayed for specifying a program handle (number). You may choose any handle (1 - 10) not presently used in the CLC card. The default handle is the next unused handle. After clicking OK, the download is executed; a bar graph displays the progress of the download.

A program cannot be run until it is activated. To activate a program, highlight it in the file list and click the **Activate** button. The active program can only be changed when the CLC is halted. The CLC is halted when in parameter mode, an error condition exist, no program is active, or after power-up before any cycle start.

To delete a program, highlight it in the file list, click the **Delete** button, and press <ENTER> or click the Yes button in the confirmation window. If a program is currently active, it cannot be deleted from the CLC card.

Selecting **Clear All** erases all programs and data from the CLC cards memory. ***See Archive*** before clearing. This process can be used to unfragment the CLC cards memory.

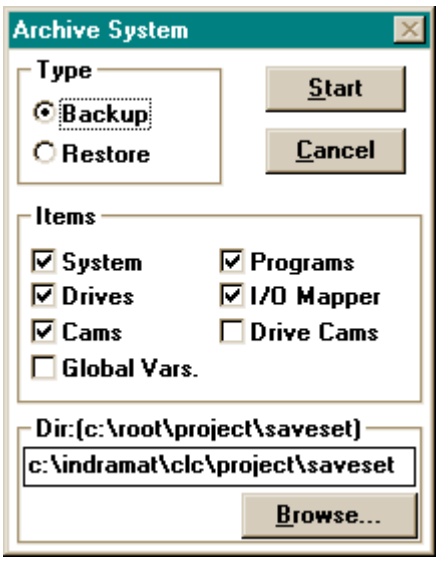
To upload a program from the CLC card to the host, highlight the file in the file list, click the **Upload** button and select the file destination and filename. A bar graph displays the progress of the upload.

Clicking on the **Data Transfer** button opens the Data Transfer dialog box. Clicking on OK copies the absolute and relative points, the events, and the integer and floating point variables from the CLC's active program to the target program selected from the CLC Program Management resident program list.

Using Data Transfer overwrites the destination program's data sets and the values are lost. If the source program's "Data Size" allocation (i.e., the size of the data sets) is larger than the target's, only the data elements within the "Data Size" allocated by the target are transferred. For example, if the active program has 75 events and the selected target program has a Data Size allocation specifying 50 events, only the first 50 events of the active program will be transferred. Data transfer is a useful tool for developing programs by incrementally testing and modifying sequential copies of a working program without the need to continuously re-input data sets for the new program.

Archive

Use of the Archive System dialog box is similar to the Transfer Point Tables dialog box. This menu item provides backup and restore functions for a CLC card. Backup programs, I/O Mapper strings, and parameters of the CLC card and active drives are saved by default to the \CLC\SAVESET directory. The **Browse** button allows the user to select another directory location. Programs, I/O Mapper equations, CLC card parameters, and drive parameters can also be selectively restored.



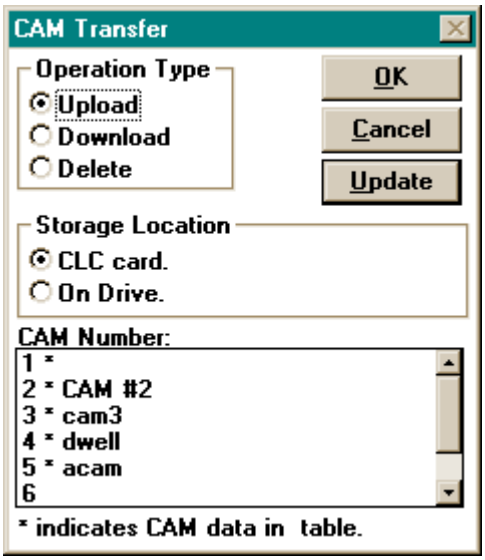
Select **Backup** or **Restore**, then press the **Start** button to transfer. If restoring, the check boxes are enabled to allow selection of kind of data to be restored. During data transfer, a bar graph will advise you of progress.

Only parameter sets of active drives(those defined in the active program) will be backup. VisualMotion does not have knowledge of other drives. Backup files are named as follows:

Drives	"drive*xx*.prm"	where *xx* is drive number
System	"system.prm"	
I/O Mapper	"mapper.iom"	
Programs	"*mmmm*.ex*n*"	where *mmmm* is program name, *n* is program handle
Cams	“cam.csv”	

Transfer Cams

This menu it is used to upload a cam table to a file for archiving, download a cam table file to the CLC card, or delete a cam table on the card. Cam table files are stored with the ".csv" file extension together with other project files in the Project Directory defined using the Setup/Configuration command. The "csv" format is a standard used by EXCEL and other spreadsheets. Each line of the file consist of two ASCII floating numbers separated by a comma, the line is finished with a <CRLF>.



The number of lines in a cam file to be downloaded can be from 10 to 1024. At the end of the download, the CLC expands the table using a 5th order polynomial to 1024 lines. When a table is uploaded from the CLC, the expanded table(1024 lines) is saved to a file.

There are 8 cams tables possible on CLC cards, these tables are dynamically allocated to conserve memory when not used. There can be two cam tables on each drive. When a table is present a "*" follows the number in the list box.

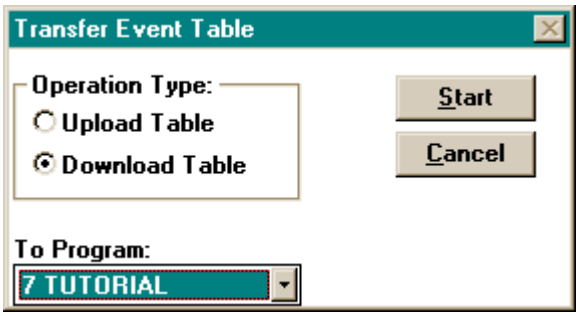
Cam tables may be downloaded or deleted at any time as long as the cam

table number is not active. If the cam is already active, the CLC responds with a communication error “Cam is already active for axis ‘x’”. To download a new table, either switch into parameter mode or deactivate the cam for all axes using it. A cam is active when it is assigned to an axis by the Cam icon.

Select the operation mode and cam number, press the **OK** button. Another dialog box will prompt for the filename on non-delete operations. During cam table transfer, a bar graph will advise you of progress.

Transfer Events

This menu item uploads and downloads event data between a program on the CLC card and a file. By default, event filenames have a ".evt" extension and are saved to the \clc\events directory. Because event data is part of the program, it is also saved when the program is saved.



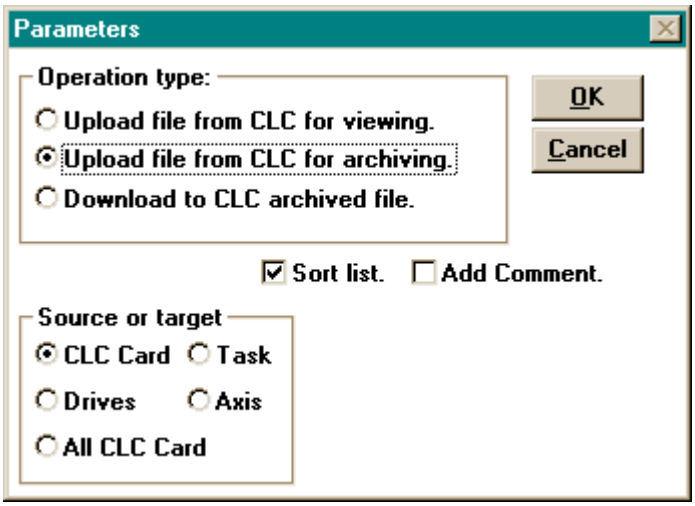
Select the operation type. The “To” or “From” Program selection (depending on the operation type) uses a combo box which contains a list of programs on the CLC card plus "Currently active". Press the **Start** button and then another dialog box will prompt the user for a filename. During the transfer, a bar graph will advise you of progress.

Transfer I/O Mapper

Use of the Transfer I/O Mapper dialog box is similar to the Transfer Point Tables dialog box with one exception. Since the CLC uses only one set of I/O mappings, a selection for a source or target is not required. I/O Mapper string files are stored with the ".iom" file extension together with other project files in the Project Directory defined using the Setup/Configuration command.

Transfer Parameters

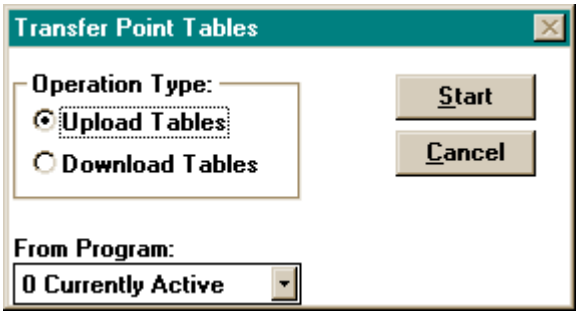
The Parameters transfer dialog box is similar to the Transfer Points dialog box with a few additional capabilities. The dialog box adds radio buttons for selecting the type of parameter set to be transferred. All CLC parameters may be transferred at one time, or parameter sets for the CLC card, Tasks or Axes may be individually transferred. In addition, the parameter set for a selected DDS-2 drive connected to the CLC may be transferred through the SERCOS communication system.



Parameters may be uploaded in one of two formats. Uploading for archiving saves the file with the ".prm" file extension in the :\CLC\SAVESET directory, with the data in the proper format for downloading to the CLC. Uploading for viewing saves the file to the same sub-directory as a text file with a ".txt" extension and may be viewed using Notepad or another ASCII text editor or file viewer. A *.txt parameter set uploaded for viewing cannot be downloaded to the CLC.

Transfer Points

The Transfer Point Tables dialog box is used to transfer the absolute and relative point tables between a resident CLC program and a point file on the Host system hard drive. Both the absolute and relative point tables are transferred at the same time.



Selecting Upload Tables and a CLC program opens a Save As dialog box permitting entry of a filename. Point files are stored with the ".pnt" file extension together with other project files in the Project Directory defined using the Setup/Configuration command.

Selecting Download Tables and a CLC program opens a File Open dialog box with :\CLC*.PNT selected by default. Clicking on OK downloads the selected file to the specified CLC program's absolute and relative point tables. A user may specify a different sub-directory and file extension as long as the file contents are in a proper format.

Attempting to download a data set larger than that specified by the Size in CLC resident program will cause an error. Acknowledging the error terminates the operation with the remaining points not transferred.

Transfer Variables

This menu item uploads and downloads program variable data between a program on the CLC card and a file. All float(Fx) and integer(Ix) program variables are saved together in one file. By default, variables filenames have a ".var" extension and are saved to the \clc\var directory. Because variable data is part of the program, it is also saved when the program is saved.

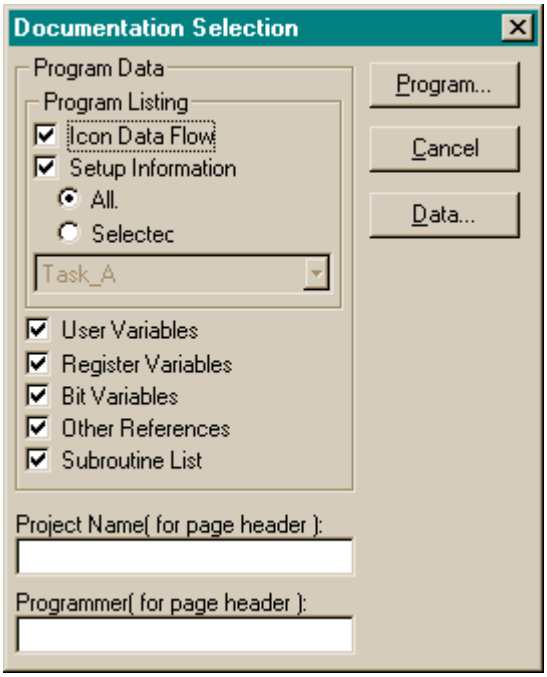
The “To” or “From” Program selection is through a combo box containing a list of programs on the CLC card plus the "Currently active" one. Select the operation type, press the Start button. Another dialog box will prompt for the filename. During transfer, a bar graph will advise you of progress.

Transfer Zones

Use of the Transfer Zones dialog box is similar to the Transfer Point Tables dialog box. System Zone files are stored with the ".zon" file extension together with other project files in the Project Directory defined using the Setup/Configuration command.

Print

Choosing "Print ..." from the File menu displays the Documentation Selection dialog box. All program data is selected by default. Checking **Icon Data Flow** provides a graphic printout of a VisualMotion Icon Language window. To print a graphic window you must have a graphics-capable printer. Text-based files, such as Text Language user programs or parameter files uploaded for viewing, may also be loaded and printed from Window's Notepad or another editing program.



Setup Information can be printed for “All” or a selected task, subroutine, and event. Selecting **User Labels**, **Register Labels**, and/or **Bit Labels** prints a list of the respective labels for the task, subroutine, or event window currently displayed by VisualMotion.

Fields for **Project Name** and **Programmer** permit adding identification to printed pages. A header on each page contains: filename, task name, date, time, and page number. Date and time are relative to the time of printout and is based on time kept in the PC. The header can be customized by filling in the edit fields of project name and programmer.

4.3 The Edit Menu

The Edit menu contains Windows editing features for icon based programs, add event functions, and editing of user, register, and bit labels.

Undo - undoes last icon edit.

Cut - cuts selected program flow to paste buffer.

Copy - copy selected program flow to paste buffer.

Paste - enables paste operation into selected program space.

Delete - deletes selected program flow from program space.

Select All - selects entire program flow of current screen.

Clear Current Task

This deletes all contents of the current VisualMotion task, subroutine, or event workspace.

Find, Find Next

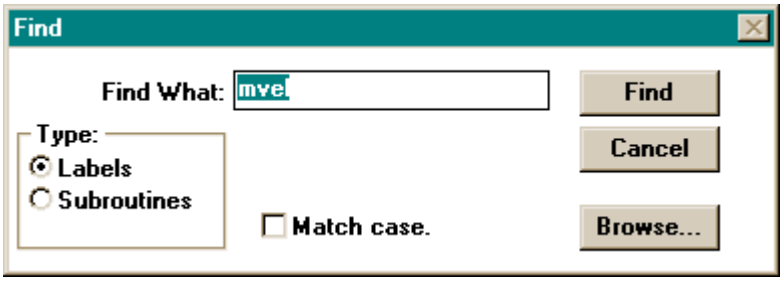
This menu item is used to locate labels or subroutine/event functions in the opened program.

To find the first occurrence of a label or subroutine:

Click label or subroutine in type grouping.

Enter the name in the Find What box. The Browse button will provide a listing of all labels or subroutines found in the opened program, one may be selected from the list.

Click the Match Case checkbox to limit search to exact match.



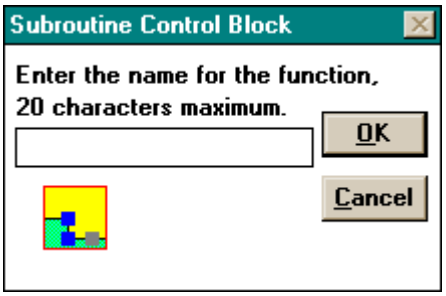
To find next occurrence of a label or subroutine:

Select Find Next in the Edit menu or press the F3 key.

Add Subroutine

Selecting Add Subroutine from the Edit menu opens a Subroutine Control Block dialog box and allows the user to write a subroutine that may be used in conjunction with a Sequencer.

Enter the name of the desired subroutine in the edit field. The name must begin with an alpha character. Pressing OK button changes to the program workspace with the name entered. This screen is not saved unless one or more icons are placed on the program workspace before changing to another. A maximum of 200 screens, consisting of tasks, subroutines, and event functions are allowed.



Add Event Function

Selecting Add Event Function from the Edit menu opens an Event Function Control Block dialog box and is the only way to write a new event function.

Event functions are similar to subroutines. However, event functions are not "called" from a program. Instead, an event function is "triggered" by the conditions (distance, time, etc.) that are specified in an event setup (refer to ***Calc icon*** in the chapter, “Icon Programming” and the description of the ***Event Table*** in the chapter, “Programming Concepts”). Note that an event function must be written before it can be assigned to an axis.

Entering a name for an event function and clicking OK replaces the current VisualMotion workspace with an event function workspace. You may then write an event function using icons and connecting lines in the same manner as when writing a task or subroutine.

Labels - User Labels

Assigning symbolic names to system resources, such as axes, drives, etc., is a very useful programming technique. Labels may also be used for absolute or relative point names, or in place of "literal" constant or variable values in expressions. For example, once assigned, the label "PI" can be used throughout a program, instead of repeatedly entering the literal value 3.14159.

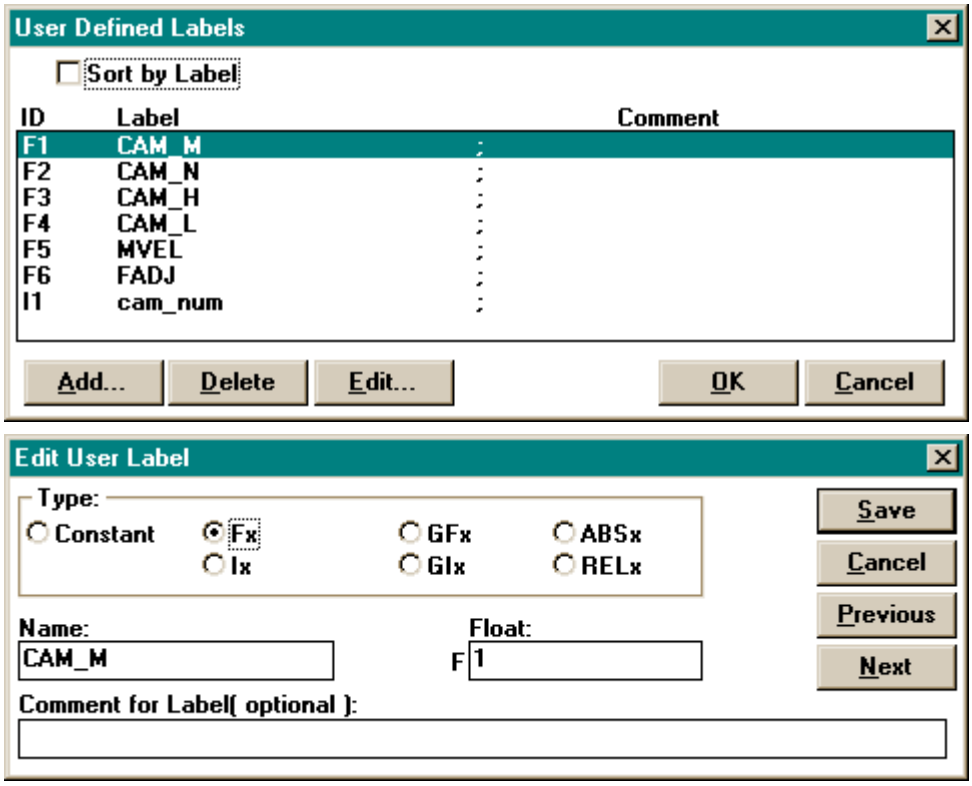
Symbolic names, or Labels, may use up to twenty ASCII characters and are case-sensitive. Blank spaces are not allowed within a symbol. Use a

printable character as a separator if it is required for clarity. For example, "next_move", rather than "next move". The first character of a symbolic name or label is not limited to an alpha character, however labels starting with a numeric will cause a 2nd pass compiler error.

The CLC compiler, used for both Icon and Text Language programming, allows the use of a literal integer value (i.e., a number such as "1" or "5") providing it is within the range of integers that are valid for the specified argument. Integers used to specify system devices, such as an axis or drive, must be within the range permitted by the complete CLC system and software installed. For example: a CLC system with eight DDS-2 drives installed can specify an axis or drive using an integer from 1 to 8. The compiler must be able to resolve a symbol used as a table index argument to an integer index within the range, or size, of the table.

User Defined Labels Dialog Box

The User Defined Labels dialog box allows you to assign an ASCII name to a value or system component, as previously described. This dialog box can be accessed from most VisualMotion dialog box data entry fields. Simply click once in the data entry field, then double-click anywhere in a non-active area of the dialog box. A User Defined Labels dialog box appears in an adjacent area of the screen. Thereafter, instead of explicitly entering a value or redefining a system component, the label can be entered by accessing the User Defined Labels dialog box and selecting the appropriate label. An optional eighty character comment field is also available for each user defined label.

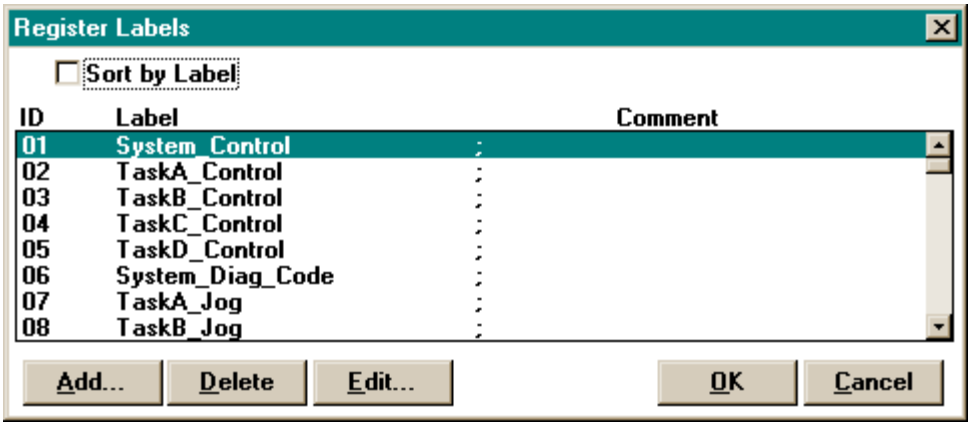


Labels - Register Labels

Selecting the Edit/Labels/Register Labels displays the Register Labels dialog box, used to provide symbolic ASCII names for CLC control status and I/O registers. The first 87 registers are reserved for CLC system use and are assigned names for system, task, DDS-2 drive, and axis use. These registers may not be modified. Registers from register 88 and above may be assigned names. If drivers for a CLC supported I/O system are installed, the associated I/O register names will be listed, typically beginning at register 100. The register list may be sorted alphabetically by name by checking the Sort by Label checkbox.

Clicking on Add or Edit (with a register selected) opens an Add/Edit Register Labels dialog box for adding or modifying a specified register name. Here the Label Name must be explicitly entered; the pop-up User Defined Labels dialog box is not available. Once modified, register labels are embedded in the motion program (the .str file) and will not be lost if the program is later transferred to a different CLC system.

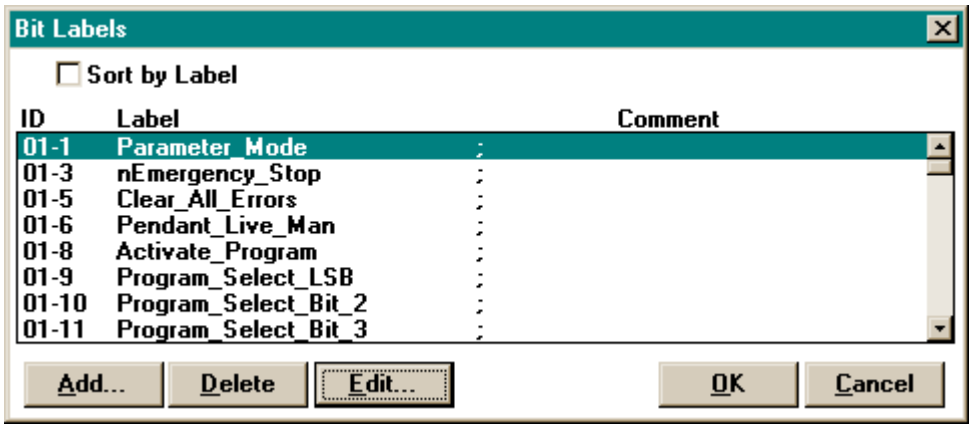
New programs are loaded with the default register names. ***Refer to the CLC Input/Output Systems chapter for the default register names.***



Labels - Bit Labels

Selecting the Edit/Labels/Bit Labels displays the Bit Labels dialog box, used to provide symbolic ASCII names for individual bits within CLC control and I/O registers. The bit label list may be sorted alphabetically by name by clicking on the Sort by Label checkbox.

The bit labels for the first 39 registers are may not be modified. Clicking on Add or Edit (with a register-bit selected) opens an Add/Edit Bit Labels dialog box for adding or modifying a specified bit label. Here the Label Name must be explicitly entered; the pop-up User Defined Labels dialog box is not applicable. Once modified, bit labels are embedded in the motion program (the .str file) and will not be lost if the program is later transferred to a different CLC system.



New programs are loaded with the default register names. ***See Chapter 2. CLC Input/Output Systems*** for the default bit names.

Import User Label File

Choosing Import User Label File from the Edit/Labels menu opens an Import Label File dialog box which permits you to import a previously saved label file into the current program. Importing a label file overwrites all existing labels in the program. User defined label files are stored by default in the *CLC* directory with a *.lbl* file extension.

Export User Label File

Choosing Export User Label File from the Edit/Labels menu opens a file Save As dialog box. Saved label files may be subsequently imported into another program allowing you to maintain standardized user labels for various categories of programs. User defined label files are saved by default in the *clc* directory with a *.lbl* file extension.

4.4 The View Menu

The View menu allows you to select a task, subroutine, or event function for display or editing. Choosing one of the menu selections loads the selected portion of the program into the VisualMotion workspace, replacing the current workspace contents.

Subroutines

The workspace may be loaded with a subroutine by using the View menu's Subroutines command. The Subroutines dialog box lists the programmed subroutines by name and displays the total number of subroutines in its title bar. Selecting an existing subroutine from the displayed list permits deleting the subroutine or replacing the current VisualMotion workspace with the subroutine for viewing/editing.

A subroutine also may be loaded to the workspace by double-clicking an existing subroutine icon to obtain the icon Subroutines dialog box. The subroutine icon's name is displayed as the default. Clicking on enter function loads the workspace with the subroutine.

Event Functions

Choosing the "Event Functions..." from the View menu displays the Event Functions selection dialog box and is used in the same manner as the View menu's Subroutines command. The View menu Event Functions is the only way to access an event function for editing or deletion. The event function must have been previously created using the Edit menu's "Add Event Function."

Zoom Out

The Zoom Out menu item is used to load the VisualMotion workspace with the parent of the presently loaded subroutine. Zoom Out, or the <F6> key, can be used to move from the back to the front of a nested subroutine queue one subroutine at a time. The queue of nested subroutines is displayed in the title bar of the VisualMotion workspace.

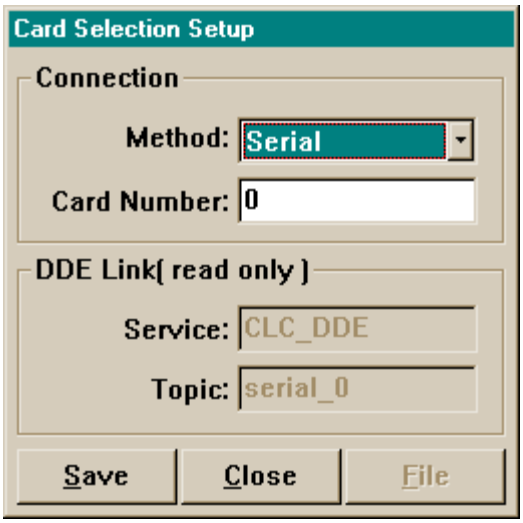
The Zoom Out command is unavailable when any of the four main tasks are loaded in the workspace or when the presently loaded subroutine was opened via the Subroutines command instead of the Enter Subroutine button of a subroutine icon. Since events functions are independent functions invoked by a specified axis motion or time-based event, they have no "parent". Therefore, Zoom Out cannot be used to return to a higher level.

4.5 The Setup Menu

Card Selection

Refer to the CLC Start Up Guide for instructions on how to use the Card Selection Setup dialog box to configure Host-to-CLC communication for your system.

After saving and compiling an icon program file, the user can now setup variables, events, point tables and zones without being connected to a CLC. To setup the off-line programming, choose “Card Selection” from the “Setup” menu. Select “CLC File” from the connection method drop down box. This tells VisualMotion® to get setup information from a file on your computer instead of the serial port, AT bus, etc. You must then select a file to use. Click on the “File” button and select the .exc file that you just compiled. Now, the programmer can enter values for variables and point tables and configure events that will be downloaded to the CLC with the *.exc* file.



Configuration

This menu item allows the user to personalize Visual Motion.

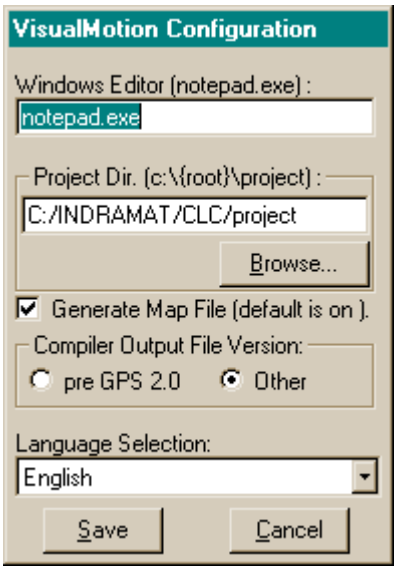
Windows Editor - Select the Windows compatible text editor to be used for non-graphical displays. The default is Notepad.exe, Write.exe is also supplied with Windows.

Project Directory - Select the directory for file storage. The default is \clc\project. The Browse button simplifies the selection of the project directory.

Generate Map File - When icon based programs are compiled a map file can be generated to be used to show program flow. On larger programs or older PCs, the generation may take several minutes. The default is on.

Compiler Output File Version - The second pass compiler can produce a file for earlier versions of the CLC card(pre GPS 2.0). ***Refer to System under the Status menu to see if the GPS version number is less then 2***. If so, choose Pre GPS 2.0. Otherwise choose Other. Features such as cams, PLS, Sequencer, local variables are not supported in the earlier version.

Language Selection: On this release only English and German are supported.



Drives

Refer to Appendix D. CLC Drive Parameter Editor.

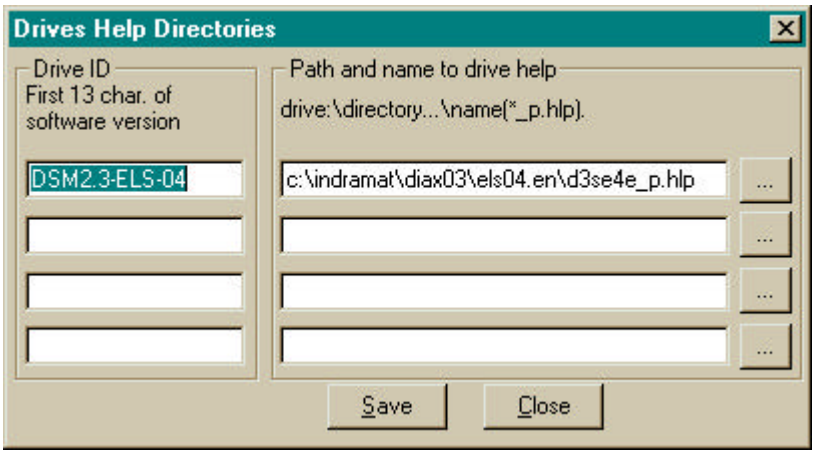
Drives Help Directories

The drive help directories is used to set the path(s) to the drive parameter help file(s). The Parameter overview screen uses this to display context sensitive help for a specific drive parameter. If you do not have the correct help files for your Drive, they can be requested from Indramat office.

The left edit field is for the first 13 characters of the drive firmware type code. It can be read from the drive editor screen or the list of drives on SERCOS ring dialogs.

The right edit field is for the complete path of the drive help file containing the parameter descriptions. The last character in the filename is 'p'.

Double click on a specific parameter to open the Drive Parameter Edit dialog box. Pressing F1 will launch the specific help topic for the selected Drive parameter.

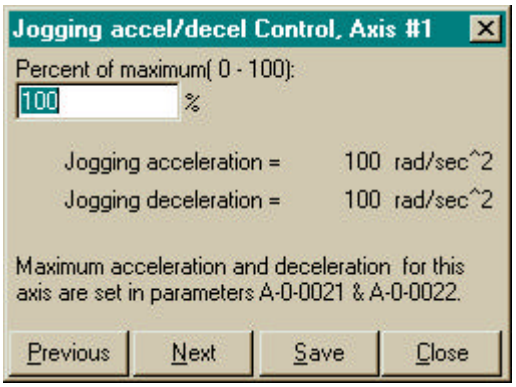


Coordinated Motion

Jogging Acceleration

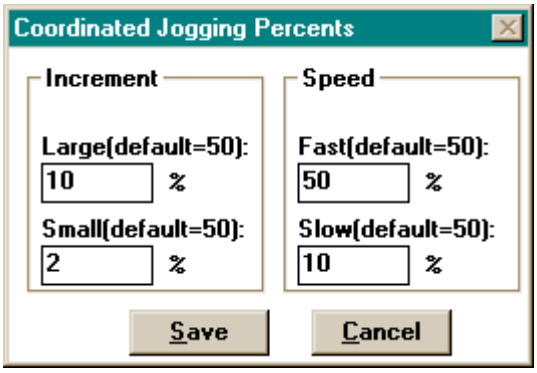
This dialog box allows users to set the jogging acceleration and deceleration for each Axis. ***See Parameter A-0-0021 and A-0-0022 for more information.***

The previous and next buttons select the different axis numbers.



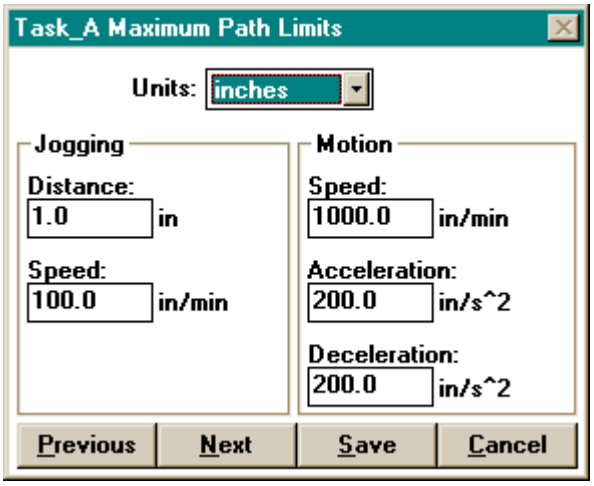
Jogging

Coordinated Jogging Percents set the increments and velocities used for fast and slow jogging. The **Increment** data area is used to set the **Large** and **Small** percentage of the maximum distance for a single-step jog operation. The maximum is defined by the axis parameter Maximum Jog Increment (A-0-0025). Similarly, the **Speed** data area is used to set the Fast and Slow jog speeds as a percentage of the maximum velocity, which is defined by the axis parameter Maximum Jog Velocity (A-0-0026). These values are stored in the following parameters: Large Increment - C-0-0052, Small Increment - C-0-0053, Fast Speed - C-0-0055, Slow Speed - C-0-0056.



Task Limits

The Task Maximum Path Limits dialog box allows speed, acceleration, and deceleration limits for coordinated motion to be individually set for each task. This dialog box permits setting acceleration and deceleration independently. The DDS-2's internal function uses the same accel/decel value for single axis non-coordinated motion.



Clicking **Save** downloads the changed values to the CLC card, without requiring the CLC to be in Parameter mode.

I/O Setup

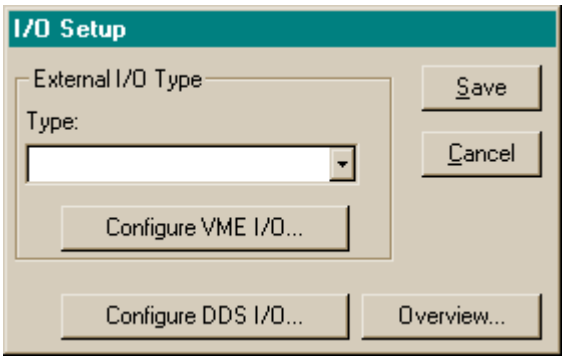
The I/O Setup dialog box allows viewing and editing of the I/O system configuration for the CLC. The general classes of I/O that can be used include:

- DDS resident DEA cards
- SERCOS based I/O
- VME cards
- DBS Interbus-S and DPF Profibus

A CLC system can use any of these classes, all, or none if controlled serially or from the VME bus by another VME device(PC, PLC). Selecting the I/O

type installs the specific device drivers for that I/O. An error will result if a I/O card is configured but is not present.

NOTE: *I/O Setup may only be altered in Parameter Mode.* ***Refer to System Control Register in the chapter "CLC input/output Systems."***



DEA Cards: Each DDS can hold up to three DEA cards, each DEA card has 15 inputs and 16 outputs. DEA cards are hardware configured to be card 1, 2, or 3. Their order in the drive rack does not matter.

DEA 28, 29 and 30 Direct Parallel I-O: These DEA option card includes 32 inputs, 24 outputs, and a Bb contact. It is connected directly to the CLC so that I-O is updated with no time delay associated with SERCOS or other I-O buses.

Up to 3 DEA cards may be connected to a CLC-D in the CCD card cage. A CLC-D residing on a DIAX02 or DIAX03 drive can have one or two DEA/C cards. The DEA-28, DEA-29, and DEA-30 are unique cards that have different bus addresses and registers allocated to them. ***See Register 400-415 for more information.***

To configure a DEA card in a DDS, select the drive and press the **Configure DDS I/O...** button. A dialog box will appear with three combo boxes. DEA cards in drives 1-8 have predefined base registers, for cards in drives 9-99 a base register must be entered. When I/O setup is complete, press the **SAVE** button to send all setup information to the CLC.

SERCOS based I/O (Lutze, etc.) is automatically configured by the CLC when they are seen on the ring. ***See Parameter C-0-2014 for more information.***

VME I/O cards supported are: Pentland MPV922, Xycom XVME-201, Xycom XVME-202, Xycom XVME-244. Select the external I/O type from the combo box. Xycom I/O cards require additional setup, press **Configure VME I/O...** button to select options. When I/O setup is complete, press the **SAVE** button to send all setup information to the CLC.

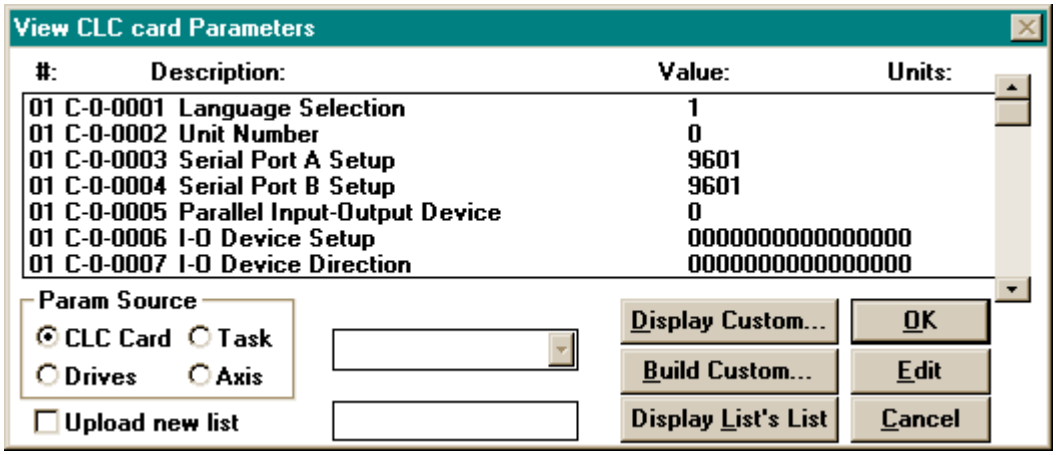
Handling of Multiple Cards: Up to 3 DEA cards and one bus interface card (DBS, DPF, or PCMCIA) can be connected to the same CLC. The bus interface cards and the PCMICA adapter share the same address space on the expansion bus. Because of this, it is not possible to have a PCMCIA interface and a DBS or DPF card connected to the same CLC. Also, the CLC

can handle only one bus interface at a time because of the system resources required by the drivers. ***See Parameter C-0-0030 and C-0-0031 for more information.***

The **Overview...** button brings up a list of currently configured I/O. This list includes the register number, drive number, module ID, I/O type and the number of registers.

Overview

Selecting Overview from the Setup menu opens a View CLC card Parameters dialog box; the initial list display is blank. This dialog box may be used to view and setup all CLC and DDS-2 user accessible parameters.



Parameters may be uploaded for display or editing by selecting one of the “Param Source” radio buttons, then clicking the OK button. Choosing CLC card allows uploading the CLC card system parameters. Choosing Drives or Axes requires the entry of a Drive or Axes Number in the enabled data entry box. Selecting Task requires choosing of one of the four CLC tasks from the enabled pull-down menu.

After the selected list has been loaded, you may scroll the list by clicking on the list scroll bar's up and down arrowheads, or clicking and dragging the scroll button. Clicking and holding (or dragging) the scroll bar button will display the number of the first viewable parameter in the display list in the upper left of the title bar.

Selecting a list entry and clicking the Edit button opens a parameter editing dialog box that allows you to change the parameter value and Update the parameter by downloading it to the CLC or DDS-2 drive.

In addition to uploading all the parameters of a set (i.e., all drive or task parameters, etc.) you may build a custom set of parameters by clicking on the Build Custom... button and opening a Build Custom Display dialog box.

When the Build Custom Display box opens, the parent View ... Parameters dialog box remains active. Selecting a list item from the parent box's list, then clicking Add in the build box adds the selected parameter to the custom parameter list. Only one parameter may be added at one time. Selecting one of the parameters in the custom list, then clicking Delete, removes the item from the custom list.

Since the parent dialog box remains active, the parameter list in the parent dialog box may be changed by re-selecting a Param Source and type. Parameters from the newly uploaded list may then be added to the custom list in the build dialog box. In this manner a custom list may be built by selecting any available parameter from any parameter list.

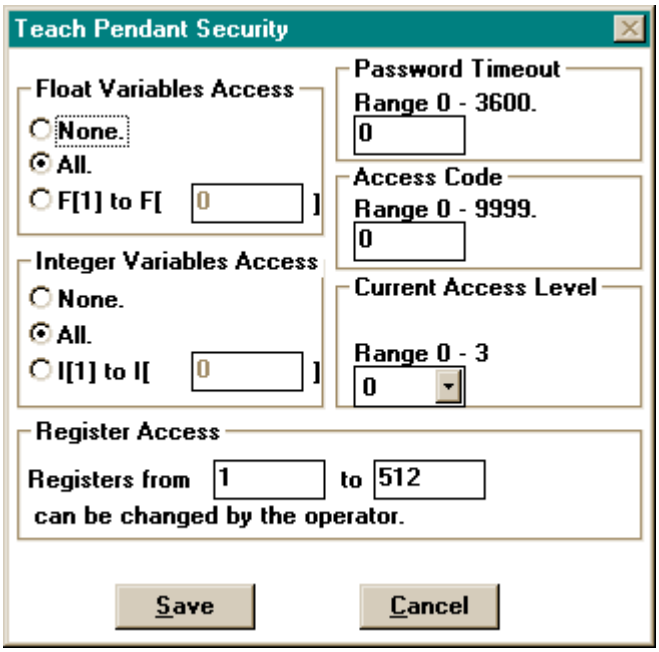
A previously saved custom list may be loaded and used as a source for building another custom list, however, it must be the first list loaded. You cannot load a custom display list once another list has been loaded without closing the Build Custom Display dialog box.

Clicking on Save pops-up a Name Custom Display dialog box permitting entry of an eight character maximum filename. Custom parameter lists are stored in the *CLC.INI* file in the *CLC* sub-directory.

Clicking on the Display Custom button opens a Display Custom List dialog box with a list of the currently saved custom parameter lists by name. Selecting a list and clicking on OK or double clicking the list item uploads and displays the parameters in the selected list.

Pendant Security

The Teach Pendant Security menu item selects the level of user accessibility for variables and registers. ***See the CLC Start Up Guide for more information on the Teach Pendant.***



CLC Serial Ports

The "**Comms Setup"** buttons permit setup of the CLC's two serial ports. Port 1 is typically used for communication with the Host system and defaults to 9600 baud. Port 2 is used to communicate with a teach pendant (default 9600 baud), if one is installed. If an ASCII "dumb" terminal is used to communicate with a CLC, the checksum should be disabled. Before saving any serial port modifications, VisualMotion requests the user to confirm the changes via a dialog box with the message, "Caution! Changing port configuration may stop communications. Continue?"

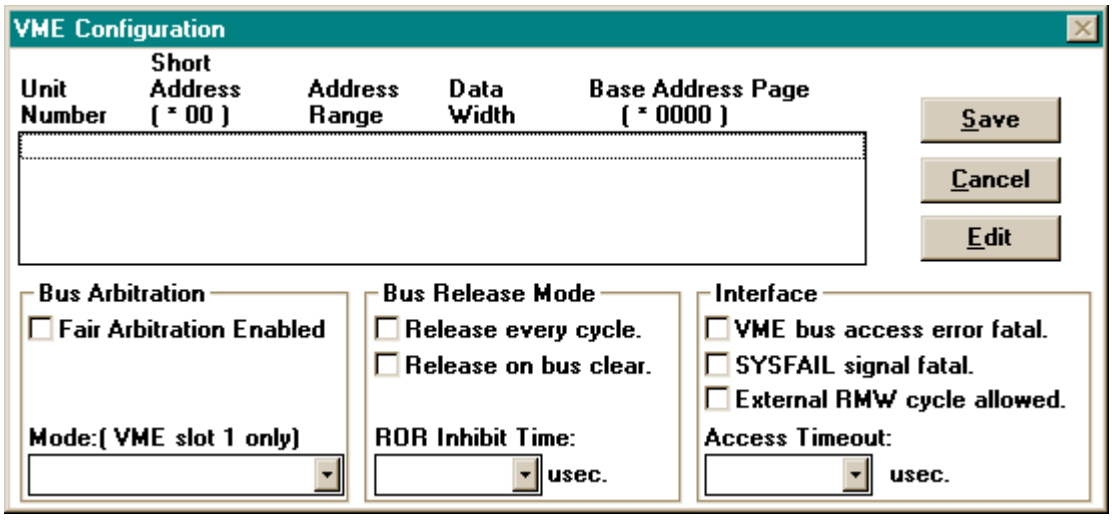
Clicking **Save** downloads the changed values to the CLC card.

VME Configure

Selecting VME Configure from the Setup menu opens the VME Configuration dialog box. CLC VME configuration beyond the standard default configuration is only required if one of the following conditions is valid for the VME system:

- There will be more than one CLC VME motion control card in the VME card cage.
- There will be more than one VME master (e.g., a CLC motion control with a VME PLC).
- There are VME I/O cards that must occupy the default CLC memory space.
- The VME system designer requires use of the CLC default memory space.

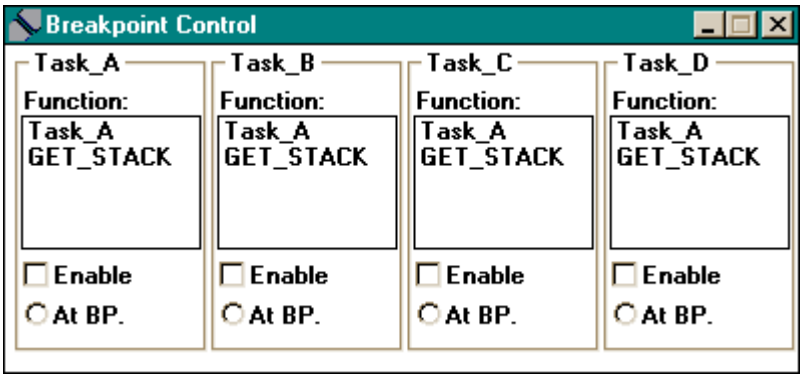
Refer to the CLC Start Up Guide Hardware Appendix for more detailed CLC-V configuration information.



4.6 The Tools Menu

Breakpoint Control

When enabled, this utility stops program flow, of a selected task, at the start of a selected functions. It is intended to be a debugging tool. Toggling the Cycle_Start_Resume(bit 6) of the task control register(2-5) of the selected task resumes program flow.



The Window is divided into four vertical grouping, one for each task. Each grouping contains a list of all functions of the active program, an enable checkbox, and an indicator of when the breakpoint was hit.

To stop at a function:

Select the function from the list-box of the task to be stopped. The list-box contains a listing of all functions, some functions are never ran from a given task (i.e. Task B is never run from Task A).

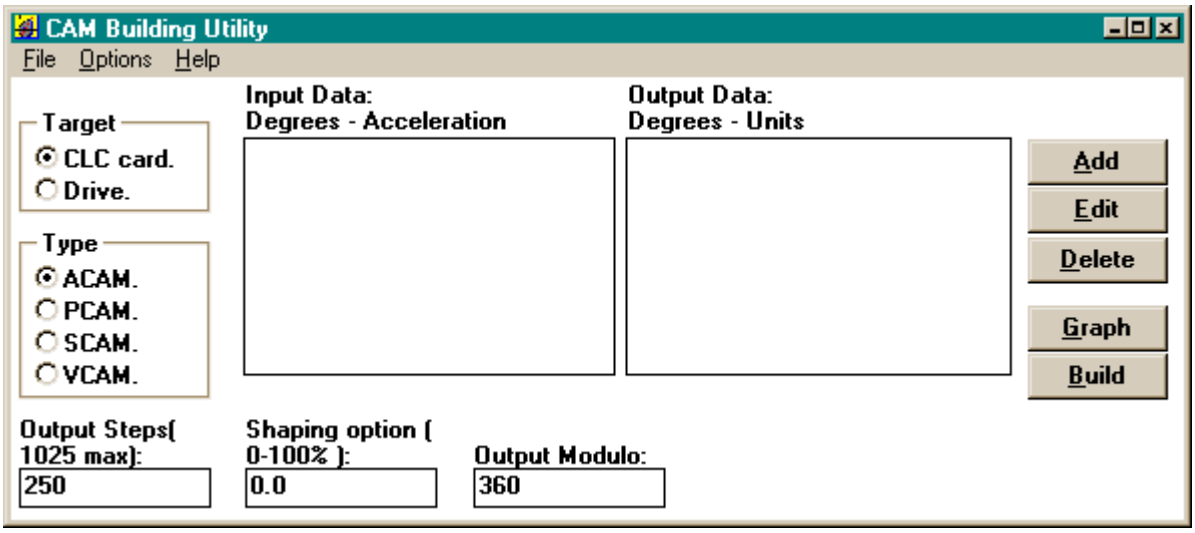
Arm it by clicking on the enable checkbox.

Wait for dot in the indicator.

Also see Register 2-5 Task Control - Single Step, Breakpoint Enable, Step Sequence Step, Step Sequence Function.

CAM Builder

The CAM Building Utility is a Windows based application used to build cam tables for Indramat's CLC motion control cards. ***See Appendix C. ELS Configuration*** for more information.



Controls:

Target - selects the format of the output data.

CLC card - the output file will be in Degrees - Units.

Drive - the output file will be in Degrees - Percent.

(*Units and Percent depend on the axis modulo*)

Type - selects the algorithm used to build output data from input data (ACAM, PCAM, SCAM or VCAM).

Output Steps - select the number of lines of output data to be built.

S Shaping (Jerk limiting) - select the amount of s shaping to be used on the velocity curve to prevent a jump in acceleration.

Output Modulo - for scaling the range of ACAM and VCAM output calculations. By default, this scaling is 360.

Input Data - list box containing input data. Double click left mouse button to edit.

Output Data - list box used to display output data.

Buttons:

Graph - creates/refreshes a graph from data in the output data listbox.

Add - add an entry to the input data listbox.

Edit - edits selected entry in input data listbox.

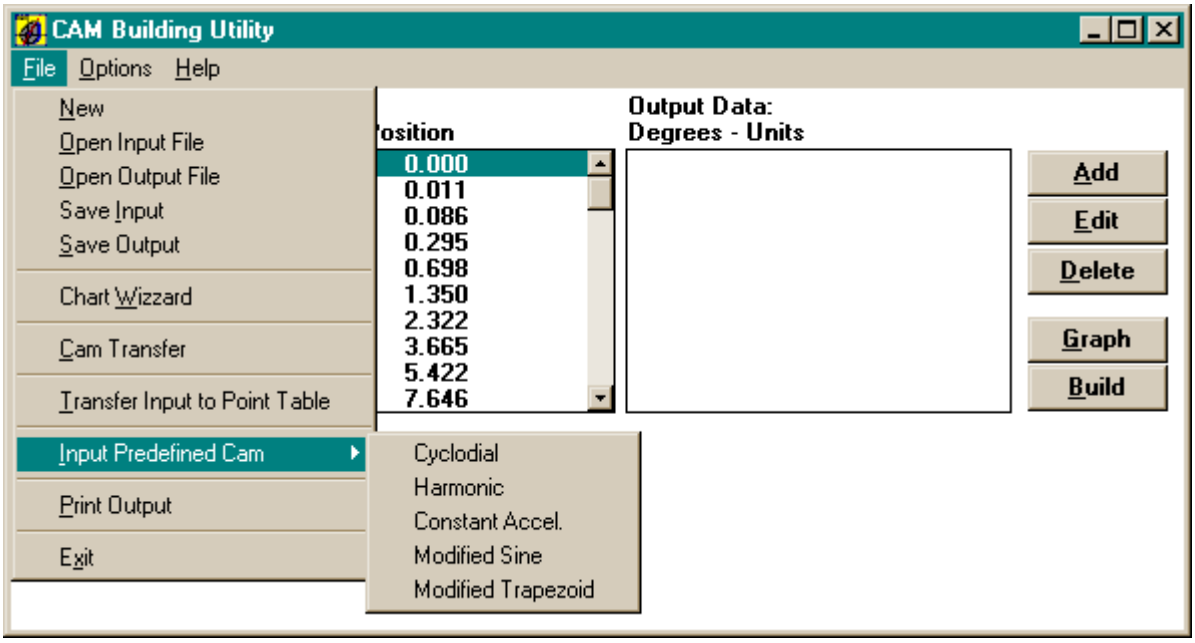
Delete - deletes selected entry in input data listbox.

Build - builds position, velocity, and acceleration profiles and displays position data in the output data listbox. Uses the algorithm selected in Build

Types to build the number of data sets chosen in Output Steps using the Input Data as input and S Shaping to modify.

Menu:

The following description outlines the Menu Items and the overall functionality of this utility.



File - getting file data, saving data to a file, printing, and exit.

New - clear input data list-box.

Open Input File - data from user selected input file is loaded into input data list-box.

Open Output File - data from user selected output is loaded into output data list-box.

Save Input - save values from input data list-box to a user selected input file

Save Output - saves values from output data list-box to a user selected output file

Chart Wizard - a series of setup screens to help setup choices on the main screen.

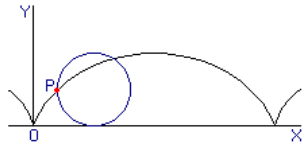
Cam Transfer - cam transfer function to move cam file to CLC or drive cam table.

Transfer Input to Point Table - This screen is used to transfers the contents of the input data list-box to points in a CLC program.

Input Predefined Cam - input data from a predefined cam file.

- Cycloidal

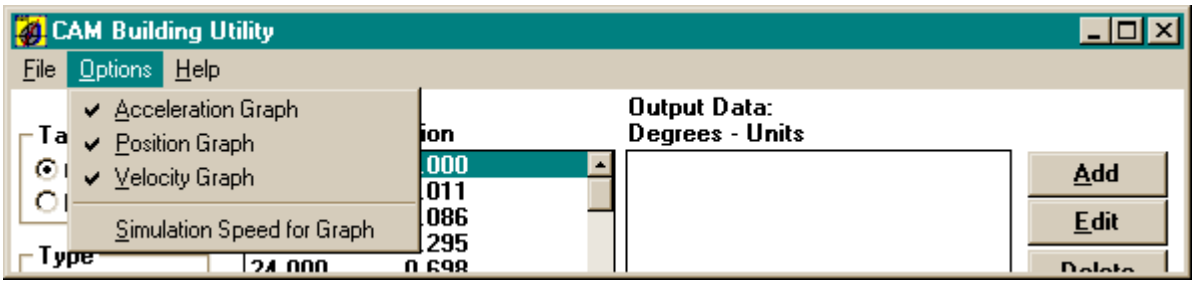
- Harmonic
- Constant Acceleration
- Modified Sine
- Modified Trapezoid



Cycloid - The curve traced by a point on the circumference of a circle that rolls on a straight line

Print Output - prints graph and output data to Windows printer.

Exit - closes cam building API.



Options - graphing and printout choices.

- *Acceleration Graph* - when checked, the acceleration graph is visible on the graph and print out.
- *Position Graph* - when checked, the position graph is visible on the graph and print out.
- *Velocity Graph* - when checked, the velocity graph is visible on the graph and print out.
- *Simulation Speed for Graph* - selects the speed used in graphing the velocity and acceleration.

Help - accessing help system and identifying product.

- *Topic* - accesses this help system.
- *About* - identifies product and date of build.

User selected input file

Input files are assumed to have the following format.

```
File identifier string.                ;first line only
master value    slave value            ;second line
master value    slave value            ;third line
          ----        ---
          ----        ---
master value    slave value            ;last line
```

Minimum and maximum number of data sets:

Build Types	ACAM	PCAM	SCAM	VCAM
Minimum Sets	2	2	5	2
Maximum Sets	1024	1024	500	1024

User selected output file

Output files are assumed to have the following format. This is commonly referred to as CSV format and is supported by Excel. Cam output files are sent to the CLC card using Visual Motion.

```
master value,   slave value        ;first line
master value,   slave value        ;second line
          ----          ---
master value,   slave value        ;last line
```

Transfer Input to Point Table

This screen is used to transfers the contents of the ‘input data list-box’ to points in a CLC program. A set of input values used for the CAM Build icon or instruction can be evaluated with the BUILD and GRAPH buttons of the previous screen. When the resultant cam is acceptable, the set of input values are transferred to the point table of the selected program on the CLC card.

Programs on Card: This combo-box list the programs on the CLC card. The list is uploaded when the screen is opened. Select the program containing the point table to be written to.

Starting Point Number for Transfer: Enter the number of the beginning point of the destination. The values of the first line of the “input data list-box” will be copied to this point. The values from each additional line of the “input data list-box” will be moved to successive point locations.

The **“OK”** button starts the transfer. Before the transfer is started, checks are made for valid point and valid range. A dialog screen will appear asking you to confirm your action. The “**Cancel**” button exits this screen.

Jogging

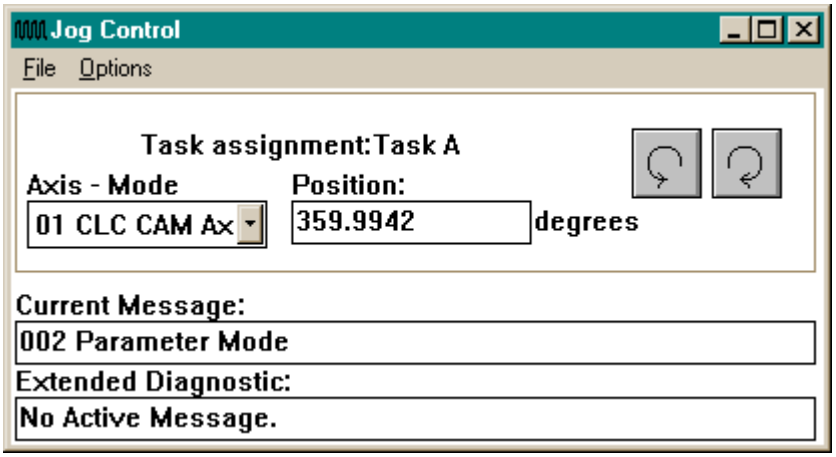
This menu item spawns an API for jogging DDS2.1, DDS2.2, DKC, and DKS drives attached to a CLC card with GPS 5.0 or later. When the CLC card is in manual mode the **Jog Control** tool allows operators to jog an axis that is in *Velocity* or *Single Axis* mode. For *Coordinated motion* all jogging functions are controlled by the Axis(*n*)_Control and Task(A-D)_Jog Registers:

Registers 11-18, 209-240: Axis Control

Registers 7-10: Task Jog Control

See Chapter 2. I/O Systems for more information

When system parameter ***C-0-0010***, bit 11 is set to 1 (*00000**1**0000000000*), jogging can also be performed in Auto Mode or when a task is running.

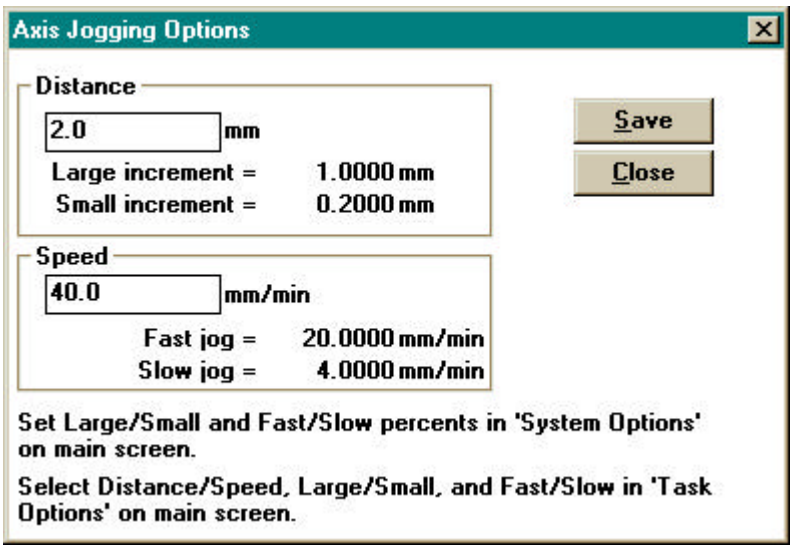


The dialog box displays the current Task assignment, Mode and Position for the active axis along with status messages and extended diagnostics. Selecting either the jog reverse or jog forward buttons respectively changes the state of bit 3 and 4 in the Axis_Control Register from 0 to 1. Deselecting these buttons will change the state of these bits from 1 back to 0.

A low-to-high (0-1) transition on these bits causes motion to start in the positive (bit 2) or negative (bit 3) direction. A high-to-low (1-0) transition immediately stops the motion. Motion is also stopped when the task mode selection changes, or when a travel limit or incremental distance has been reached.

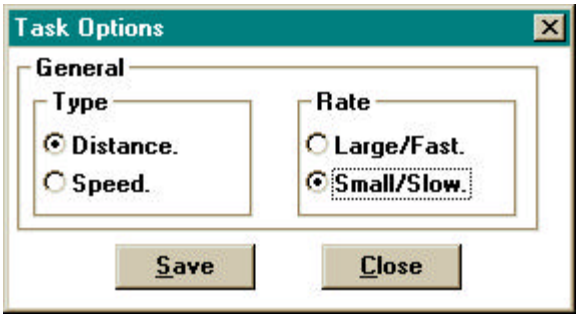
Axis Options

Selecting **Axis** from the **Options** menu will open a dialog box which allows the user to input the maximum jog distance and speed. The maximum distance value is stored in parameter (***A-0-0025***). The maximum speed is stored in parameter (***A-0-0026***). The large/small increment values and the fast/slow velocities are calculated according to the values entered in the System Jogging Options selection described below.



Task Options

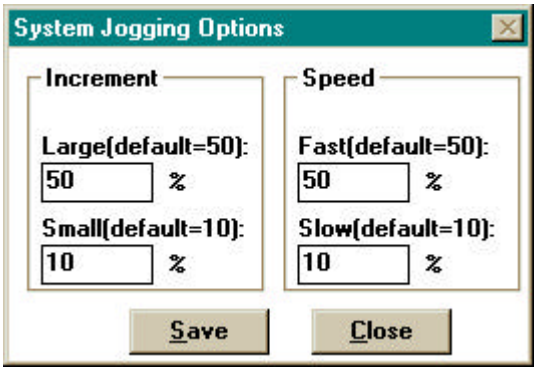
Selecting **Task** from the Options menu will open a dialog box which allows the user to select the jog type and rate. The **jog type** can be distance (incremental) or speed (continuous). The type of jog takes effect when the next jog is started with a transition on the jog forward or reverse buttons. When **distance** is selected the jogging motion will stop after the large or small travel limit is reached. When **Speed** is selected the jogging motion will continue until the jog button is deselected or the travel limit is reached. ***See Appendix D. Drive Parameter Editor-Drive Limits***.



When distance is selected The jog rate selection is **Large** or **Small**. When speed is selected the jog rate selection is **Fast** or **Slow**. ***Refer to Bit # 6 in the Task Jog Control Register*.**

System Options

The **System Jogging Options** dialog box is used for setting the increments and velocities used for fast and slow jogging. The **Increment** data area is used to set the **Large** and **Small** percentage of the maximum distance for a single-step jog operation. The maximum is defined by the axis parameter Maximum Jog Increment (***A-0-0025***). Similarly, the **Speed** data area is used to set the Fast and Slow jog speeds as a percentage of the maximum velocity, which is defined by the axis parameter Maximum Jog Velocity (***A-0-0026***).



These values are stored in the following parameter locations:

Large Increment - **C-0-0052**
Small Increment - **C-0-0053**
Fast Speed - **C-0-0055**
Slow Speed - **C-0-0056**

Single-Axis Mode Jogging

Position Limits Enabled

When a jog forward is started, the CLC sets the target position to the positive travel limit. When a reverse jog is started, the target position is set to the negative travel limit. When the jog is stopped, the target velocity is set to zero, but the target position remains at the travel limit.

Position Limits Not Enabled - DIAX03 Drives version 04 and greater

On DIAX03 drives version 4 and later, the drive is switched to velocity mode. The ramps are generated internally by the drive using the axis jog acceleration parameter **A-0-0023**.

Before performing single-axis positioning using the axis_move command, it is necessary to execute the els_mode command to switch the drive back into single-axis mode. A cycle stop followed by a cycle start will also reset the drives to single-axis mode.

Position Limits Not Enabled - Other Drives

The drive remains in single-axis mode. The CLC will continually increase or decrease the target position by a small amount to keep the drive moving, until the jog is stopped. It sets the acceleration to a value high enough so that the drive does not decelerate. The jog acceleration parameter will be used only if it is lower that this value.

Velocity Mode Jogging

DIAX03 Drives version 04 and greater

The drive generates the ramps internally based on the programmed acceleration, as described above in 'position limits not enabled'.

Other Drives

The ramp selection in parameter ***A-0-0004*** bit 9 determines if the programmed acceleration is used. The CLC generates a ramp if ramping is enabled, otherwise the velocity is immediately stepped to the programmed value.

Other Changes Since pre-GPS 5.0

When a continuous jog is stopped on a DIAX03 drive, the drive remains in an AF state, and the CLC commands it to zero velocity using the programmed acceleration.

If DIAX03 ELS-04VRS is used, there is always a ramp on the velocity as long as its programmed acceleration is nonzero. If a step is needed, the jog acceleration should be set a large value.

Refer to the following parameter and register descriptions for more jogging information:

C-0-0042	World Large Increment
C-0-0043	World Small Increment
C-0-0045	World Fast Jog Speed
C-0-0046	World Slow Jog Speed
C-0-0160	Virtual Master Maximum Jog Velocity
T-0-0025	Maximum Jog Increment
T-0-0026	Maximum Jog Velocity

Registers 31-38; Axis(n) Status - bit 2 jogging fwd, bit 3 jogging rev.

Oscilloscope

For information on the Oscilloscope function ***See Apendix D. CLC Drive Parameter Editor.***

VisualMotion32, CLC_DDE Release6

These shortcuts have been provided in order to access other programs and utilities from within VisualMotion. These links are automatically done in the Windows95 registry during installation.

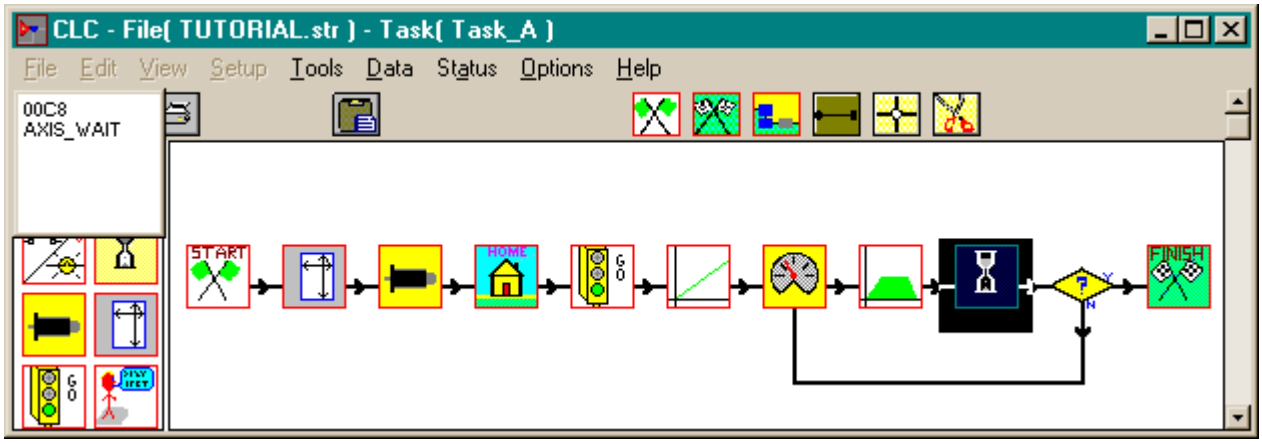
Show Program Flow

This menu item highlights the currently executing icon. Many icons operate

quickly and may appear to be skipped over. Program flow for subroutines require the operator to identify the task controlling the subroutines.

Show Program Flow utilizes a map file generated at compile time to tag the screen location of an instruction. If this map file is not found, or if the icons have been moved, added or deleted since the time the program was compile and downloaded, erroneous program flow may appear.

During Show Program Flow other menu items are disabled, selecting Show Program Flow a second time enables the other menu items and removes the highlighting.



4.7 The Data Menu

CAM Indexer

Refer to the CAM Indexer Icon. for information about the CAM Indexer Runtime Utility.

Events

Events are used to execute an event function on time, position, angle, or I/O state conditions. Each event has: {status, type, direction, distance or time, event function, and message} fields. The View Events Table dialog box permits viewing and editing of the event table of a program that has already been downloaded to the CLC card. The dialog box automatically uploads and displays the contents of the event table for the currently active program on the CLC. ***See Chapter 5. Programming Concepts*** *for more information.*

To edit an event, select it in the list-box, and press the EDIT button, or double-click on the list item. In either case, another dialog box will appear with buttons and edit fields. Change one or more fields and press SAVE button to send to CLC. Editing a selected CLC resident program's event table is one method for associating an event table entry with an event function, and specifying how the function is invoked by the program.

Events for coordinated motion are related to the beginning or end of a path segment as a percentage of the total path distance or time on the path. Events for single axis motion may be set to take place at an absolute distance from the beginning or end of the axis move. A cyclic event may be setup to be triggered by a continuously repeating timer or as an angular position on an axis using rotary motion.

Task Input Transition triggers an event on a low to high transition of a Task Control Register interrupt event bit . The I/O mapper is used to map a specified I/O register condition from the I/O register to bit 9 in the appropriate Task Control Register. The approximate latency is <= 2 milliseconds.

VME-based CLC systems may trigger events using the VME Broadcast Event or VME Short Address (mailbox) Event. These triggers are typically used to trigger or synchronize events on other CLC cards in a multi-card CLC system.

Feedback Capture uses the Probe capability of DDS-2 drives to trigger a CLC event based on a positive or negative transition of the drive's Probe 1 or Probe 2 input.

To obtain a list of a program's CLC resident events, you can pop-up an Event Functions dialog box with a list of the events. Place the focus within the "Optional 'Event Function' to run" data entry box, then double-click on a non-active area of the Edit Event Values dialog box.

After setting up an event, download the event configuration to the CLC by clicking on the Save button. The Previous and Next buttons permit editing and downloading a series of events without having to leave the Edit Event Values dialog box. Once an event function is associated with an event ID in a program's event table, an event icon is used to "arm" or "disarm" events, or to suspend program execution until the event is done.

Field Bus Mapper (CLC-D Only)

The Field Bus Mapper is designed to create data interchanging bridges between the CLC card and a Field bus slave board. Data can be mapped in both directions and specific mapping lists can be saved and opened as a file, or downloaded and uploaded from the CLC card.

The available Bus Types include: Interbus-S, Profibus. (DeviceNet, and CANopen are in development.)

The data types that can be mapped include:

Variables: Integer, Global Integer, Float, Global Float

Parameters: Axis, Card and Task

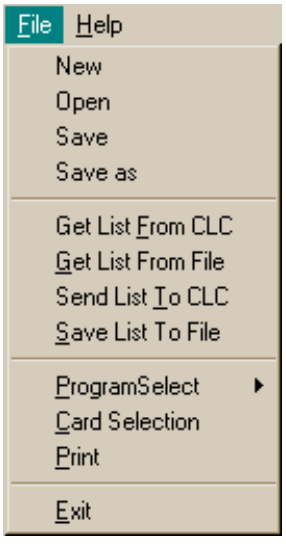
Registers

Create and Process Object Mapping List

A combination of data types can be included in the Object Mapping List. Data types are listed in their direct ASCII Communication Protocol format. ***See Direct ASCII Communication.*** To add a data type to the list following these three steps:

1. Select bus options including bus type, channel and source/destination object.
2. Select mapping direction, CLC to Master or Master to CLC.
3. Select CLC options which include data type and Data Source to be mapped.

The File Menu contains a list of options for various Field Bus configurations:



New: Clears current Object Mapping List in order to build a new one.

Open: Opens a Mapping List that has already been created and saved as a file.

Save: Saves the current Object Mapping List as a file.

Save as: Allows the user to save a file under a different name or location.

Get List From CLC: Same as Get From CLC Button.

Get List From File: Same as Get From File Button.

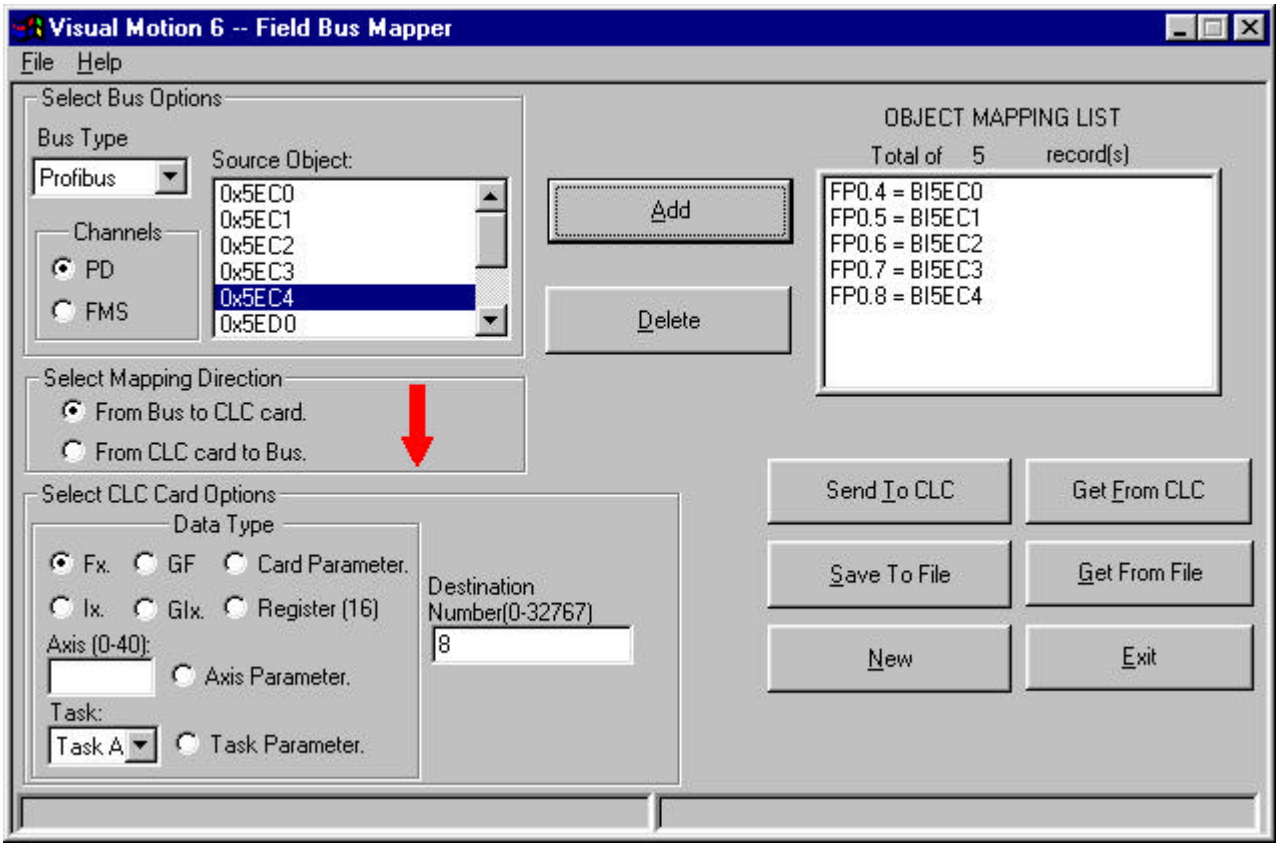
Send List To CLC: Same as Send To CLC Button.

Save List To File: Same as Save to File Button.

Print: Prints the Object Mapping List.

Card Selection: Opens the Card Selection Dialog Box. Same as under the Setup Menu in VM.

Exit: Closes the Field Bus Mapper



After adding data types to the Object Mapping List it can be edited by double clicking on an item in the list. The **Add** and **Delete** buttons will be replaced by **Insert**, **Replace**, **Delete** and **Back To Add** buttons.

When a Variable is selected a dialog box will open that lists the available programs on the CLC card. Select the program which contains the variable that you want to map and a list of applicable variables will appear in a selection box.

When a Parameter (Card, Task or Axis) is selected a dialog box will open that lists the available parameters. Before select Axis or Task parameters be sure to enter to specific number or letter designation.

When a Register is selected a dialog box will open that lists the available programs on the CLC card. Select the program which contains the Register(s) that you want to map and a list of Default Registers will be displayed in a selection box.

DP/PD Channel Configuration

This utility allows users to configure the cyclic channels (PD or DP for Profibus, Interbus-s) of the field bus slave board. To bring up this utility select the Bus Type in the main Field Bus Mapper dialog box. A *Configure Slave* button will appear above the source object list. This button launches the following dialog:

The Device (or Slave) Address (C-0-2630) can range from 2 to 126. Any number over this range will be rejected (For Profibus only).

The maximum length of the PD/DP Channel (C-0-2634) is 16 words that include the length of the parameter channel and the length of the bus configuration objects.

The length of the Parameter Channel (C-0-2631) can't be larger than the length of the PD/DP channel. Entering 0 will disable the parameter channel.

The Multiplex Channel control word consists of three elements:

- I/O (Multiplex input, output or both)
- Width of Multiplex (Highest PD group, range from 0-F)
- Start Offset (Defines the Multiplex start pointer in the bus configuration list.

The available Objects list displays the Base object (for example: 032_3_B_5EF0 used for 32 bit Multiplex mapping) and the Single object (for example: 016_1_S_5F92 used for single 16 bit mapping). The Base object is the one that is allowed to be used as the first object of the multiplex objects.

The length of the Bus configuration IN and OUT lists have to be the same. If not, an error message will be displayed.

All data entered in this dialog box will be verified after the OK button is selected. If the data is correct, it will be saved to the appropriate Card Parameters.

The available source objects in the main Field Bus Master dialog box will include the Multiplexing selection. These objects are based on the information entered for the DP/PD Bus Configuration IN/OUT lists.

I/O Mapper

Overview

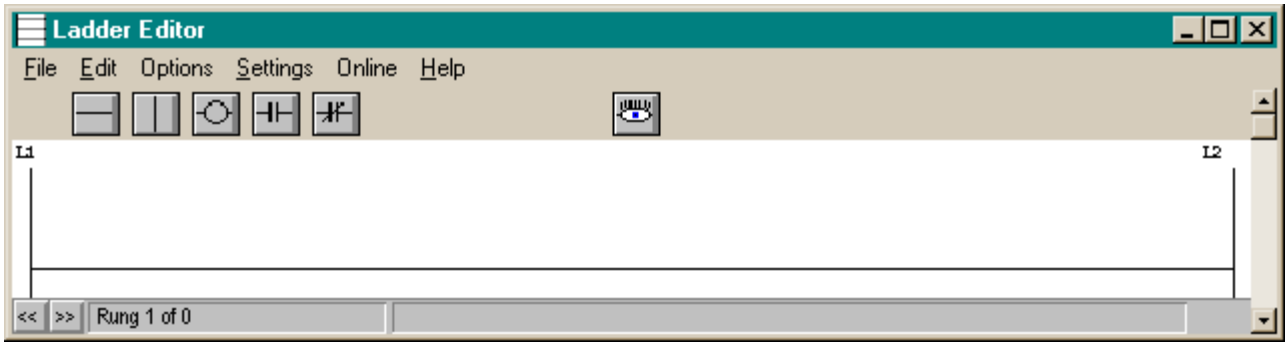
The CLC I-O mapper allows manipulation of I-O registers using Boolean strings and an optional ladder logic interface. This can be used to map physical I-O bits to the control and status registers. Simple PLC functionality is included, with AND, OR, and NOT functions. The I-O mapping program interpreter executes all equations every scan time (4ms or 8ms) in all modes, independent of the User Tasks. This allows a deterministic response to critical I-O.

Storage and Compilation

An I-O mapper string file (.IOM) can be generated from the ladder logic interface or entered using a text editor, and saved on the PC. This file is then sent to the CLC, where it is compiled to an executable format.

The mapper is stored as Boolean equations. On the CLC, comments can take a large amount of SRAM storage. The ladder interface shows the total space consumed before the file is sent to the CLC, and the user can optionally disable storing comments on the CLC.

NOTE: *The I/O Mapper may only be altered in Parameter Mode. Refer to Chapter 2. CLC Input/Output Systems.*



File Menu

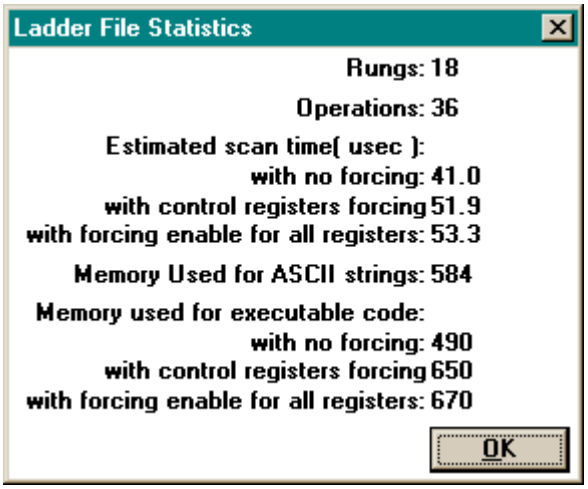
New - Opens a new Ladder Editor - I/O Mapper File

Open - Opens an existing I/O Mapper file (*.iom)

Save - Saves a named I/O Mapper to a file.

Save as - Saves an I/O Mapper to a file, filename is prompted for.

Current Statistics - Lists the statistics of the open I/O Map file.


Ladder File Statistics
Rungs: 18
Operations: 36
Estimated scan time(usec):
with no forcing: 41.0
with control registers forcing 51.9
with forcing enable for all registers: 53.3
Memory Used for ASCII strings: 584
Memory used for executable code:
with no forcing: 490
with control registers forcing 650
with forcing enable for all registers: 670
OK

Upload Strings - When Uploading an I/O Mapper from the CLC card, the Ladder Editor will prompt the user before overwriting any existing rungs.

Download Strings - When Downloading an I/O Mapper to the CLC card, the Ladder Editor will prompt the user before overwriting any existing rungs. *The CLC card must be in parameter mode.*

Print Output - Prints all or individual components of the I/O Mapper file which include rungs, comments, strings and cross references. A project and programmer name can also be sent with the print job.

Edit Menu

Add rung - Adds a rung to the end of the I/O Mapper.

Delete rung - Deletes the currently displayed rung from the I/O Mapper.

Insert rung - Adds a rung in front of the currently displayed rung in the I/O Mapper.

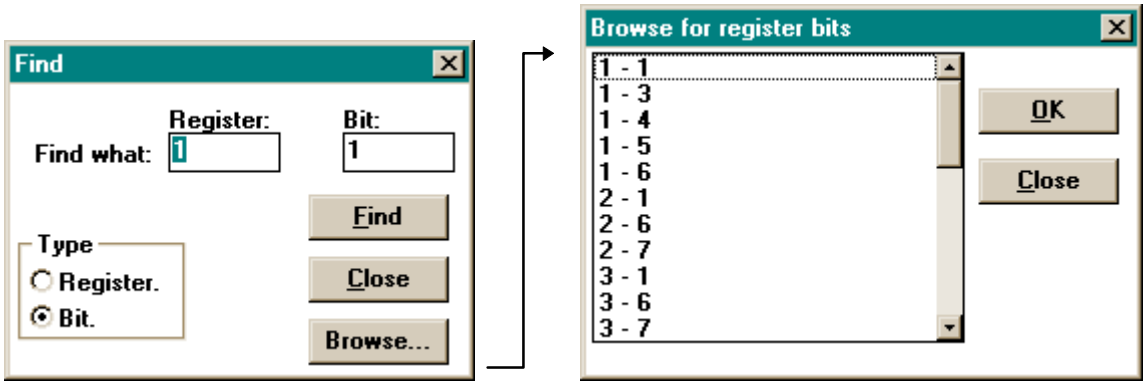
Copy rung - Copies the currently displayed rung to a clipboard.

Paste rung - Pastes (overwrites) the contents of the clipboard in the currently displayed rung.

Check - Checks the rung for missing contacts, coils, shorts and non-continuos lines.

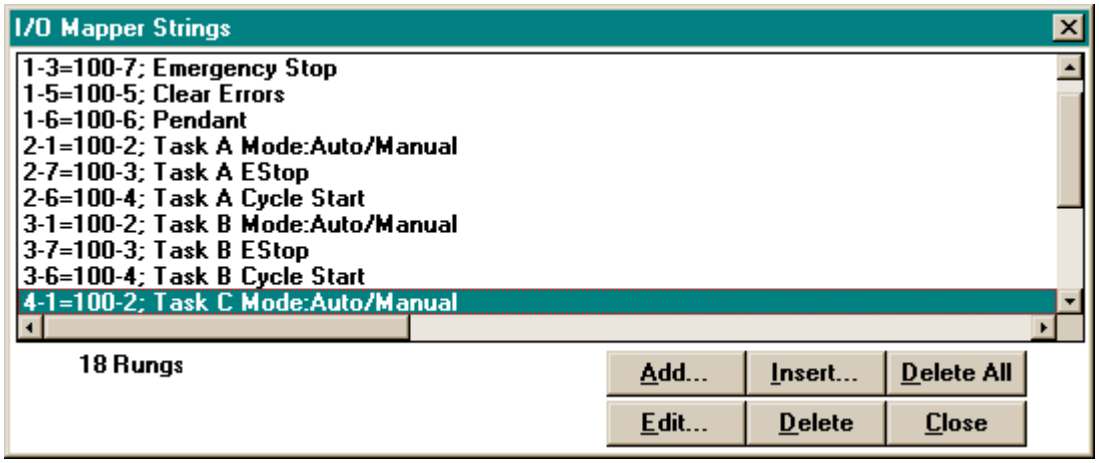
Find…- Finds the rung which contains a specific Bit within a Register.

Find Next - Finds the next occurrence of the above within the I/O Mapper.

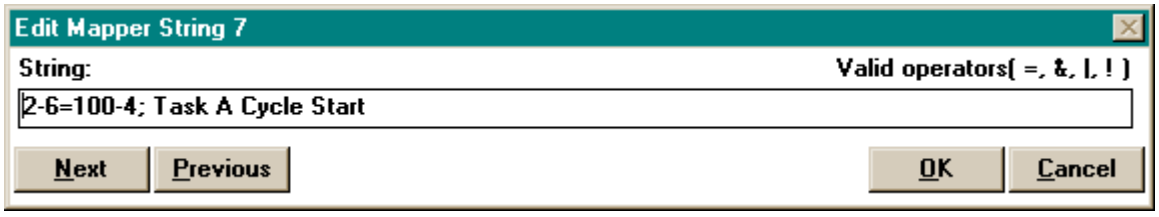


Options Menu

Display Strings - Displays each rung as a Boolean argument represented by an ASCII text string.



Double clicking a string in the displayed list or selecting one and clicking the Edit button opens an Edit Mapper String dialog box. The Edit Mapper String dialog box allows editing of I/O Mapper strings resident on the CLC.



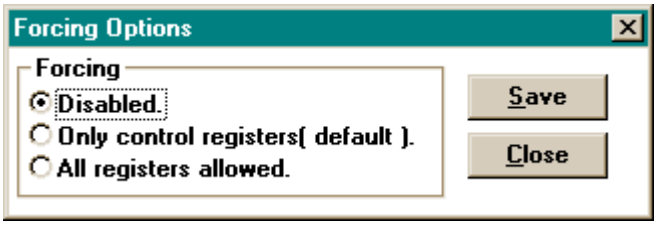
For proper use and limitations of I/O mapping *see Chapter 2. CLC Input/Output Systems - I/O Mapper*.

A semi-colon at the end of an I/O Mapper string will allow comments to be entered. After adding or editing one or more strings, the Save button may be used to download changes to the CLC; a bar graph displays the progress of the download.

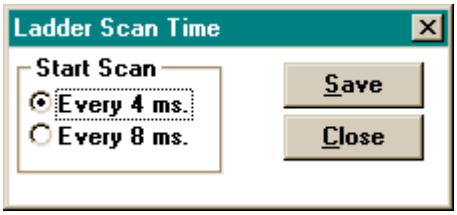
Settings Menu

Card Selection - launches the same dialog box accessed under the VisualMotion Setup Menu.

Forcing - Sets the forcing options for the I/O Mapper. Choose Registers under the Data menu to change or force bits.

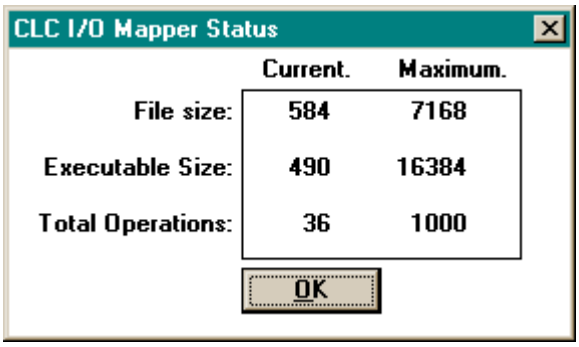


Scan Time - The I/O Mapper equations can be executed every 4 or 8 milliseconds, and therefore must be kept to a minimum to allow sufficient time for motion tasks.



Online Menu

Status - Contains Current and Maximum values for File size, Executable size, and total I/O mapper operations.



The following Function Keys are used with the Ladder Editor:

F1 - Help

F4 - Find Next

F5 - Previous Rung

F6 - Next Rung

F7 - Show Status

Right Mouse Button

Delete - Deletes the contents of the square pointed to.

Edit - Open a dialog box to change register, bit, or contact type.

Copy - Copies the contact or coil pointed to into memory.

Cut - Copies the contact or coil pointed to into memory and clears it.

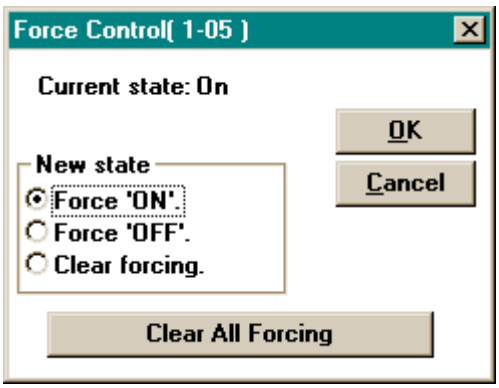
Paste - Pastes a contact or coil from memory into current rung.

Undo - Undo the last edit operation.

Insert column - Shifts contents of column pointed to, to the right. Existing lines are extended. Error if right most contact column in use.

Insert row - Shifts contents of row pointed to down one row. Existing lines are extended. Error if bottom row is not empty.

Force term - Opens dialog box which allows the user to force "on" or "off" the selected register bit. Clear forcing clears the selection. "Clear All Forcing" clears all within I/O Map.



Double click Left Mouse Button - Edits contact or coil pointed to.

PID Control Loops

PID Overview (See also *PID Icon*)

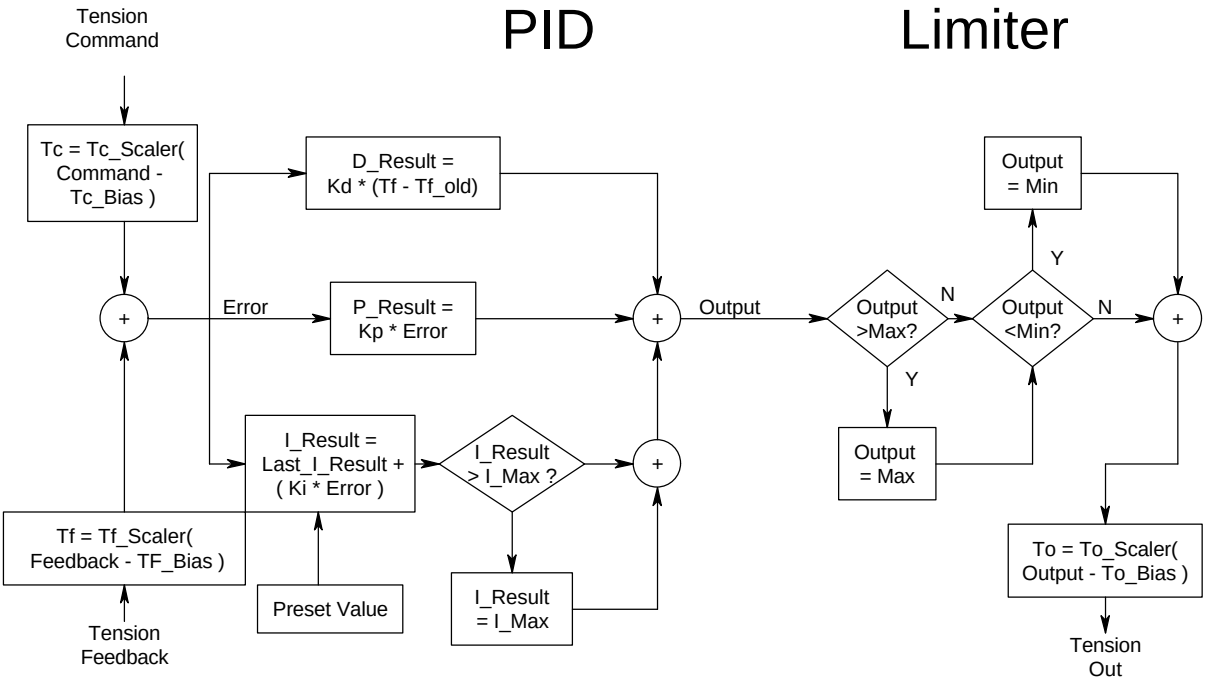
CLC programs, GPS 5.00 and later, can have up to 10 PID loops. The PID loops are added to the program with the PID icon or PID/INIT textual command. PID loops can be monitored and tuned using the PID Monitor.

The PID instruction is activated at program activation with SERCOS ring in phase 2 or greater and it's control register "PID Enable" (bit 5) set. The tasks do not need to be running.

The CLC can include up to 10 PID (Proportional, Integral, Derivative) control loops with each program. These PID's are parameterized with program variables or registers, and have a minimum update rate of 8ms. A choice of optional filters (Low pass or Butterworth) may be applied to the feedback signal.

PID Instruction.

The PID instruction configures a PI or PID control loop. The set point, feedback, and output variables can be registers, integers, floats, or parameters; appropriate conversions are supplied. Control factors(Ki, Kp, Kd, Last_I_Result_Preset) and limits(min., max.) can be constants or variables. Minimum loop update time is 8 milliseconds. In operation, the PID instruction only needs to be executed once in the program flow. The label for the PID loop is it's control registers label.



Control Register

Bit 4 Enable Preset - If set, loads integral preset, on program activation or when loop enabled by bit 5.

Bit 5 Enable PID - If set, enables loop, clearing it, disables it. This bit is checked every SERCOS update.

Status Register

Bit 4 Enable Preset - Acknowledgment of control register preset bit)4).

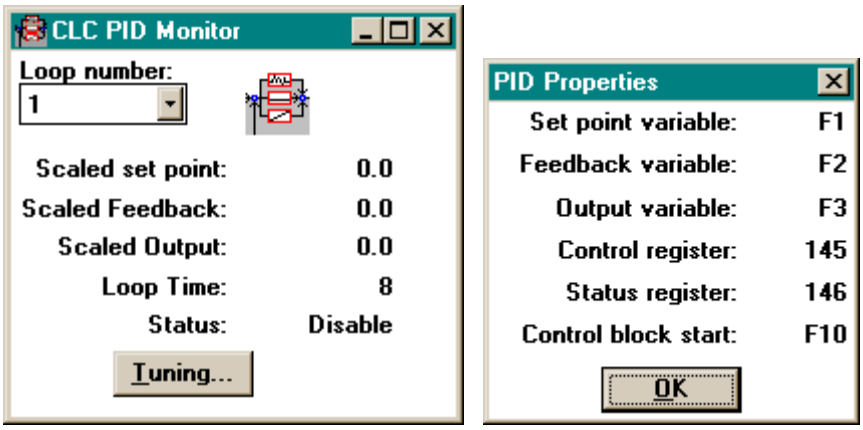
Bit 5 Enable PID - Acknowledgment of control register enable bit(5).

Block of 20 program float variables per PID loop (Fx). (Type 1 PID usage)

F20	Command scaler value, default 1.0.	4	
1	Command bais value, default 0.0.	4	
2	Feedback scaler value, default 1.0.	4	
3	Feedback bais value, default 0.0.	4	
4	Kp value, default 1.0.	4	
5	Ki value , default 0.0.		4
6	Kd value, default 0.0.	4	
7	Ki limit value, default 0.0.	4	
8	Minimum output value, default -10.0.		4
9	Maximum output value, default 10.0.	4	
F30	Preset value, default 0.0.	4	
1	Output scaler value, default 1.0.	4	
2	Output bais value , default 0.0.	4	
3	Feedback cutoff frequency (Hz), default 0	4	
4	Feedback filter type, default 0	4	
	0=None		
	1=First order low-pass		
	2=Second order low-pass		
	3=Third order Butterworth		
	4=Second order Butterworth		
	5=Third order Butterworth		
	6=Velocity tracking 2nd order		
	7=Accel ramp tracking 3rd order		
5	reserved	4	
6	reserved	4	
7	reserved	4	
8	reserved	4	
9	reserved	4	

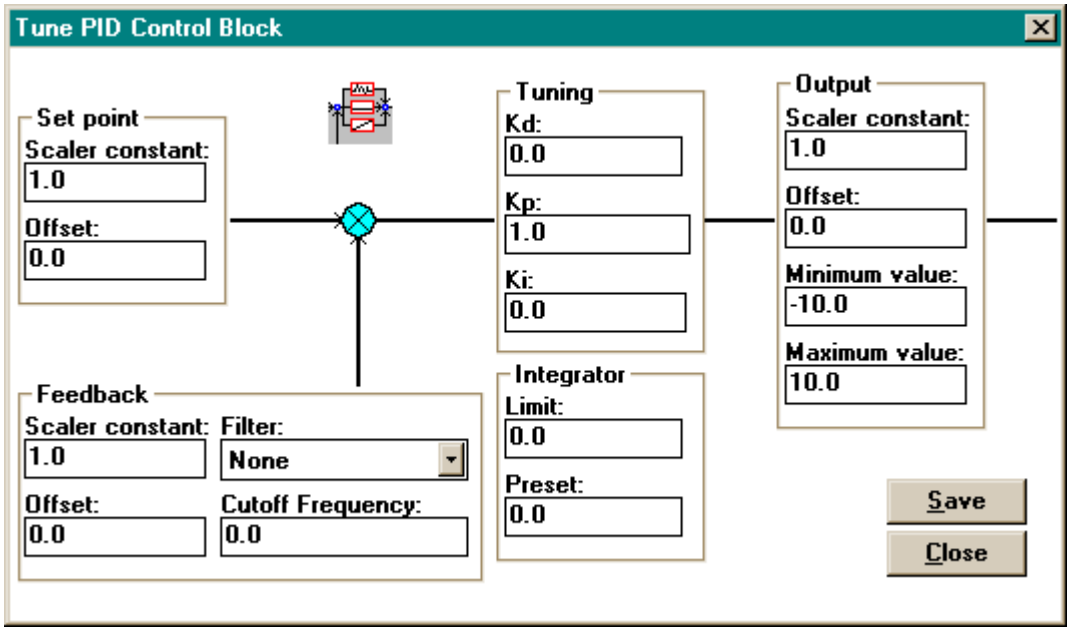
PID Monitor - Overview

This menu item spawns a API for monitoring and tuning the PID of the active program on the CLC card. CLC programs, GPS 5.00 and later, can have up to 10 PID loops. The PID loops are added to the program with the PID icon or PID/INIT textual command.



The initial screen displays a list of PID loops of the active program, and the set point, feedback, output, loop time, and status of the selected loop. All fields, except loop time are continuos updated. PID loops are activated when a program is selected, loops of other programs are not viewable. Pressing the right mouse button displays a menu list of: About, Update, and Properties.

A Tuning Button displays a screen of parameters for editing.



The tuning screens is used to adjust the selected PID loop while monitoring its values on the main screen. This screen has several grouping for scaling and adding offset to the process variables. A grouping adjusting the Kd, Kp, Ki, integral preset, and integral limit. The output grouping also has min, max limits for it. The feedback has an optional digital filter to condition the signal. The feedback filter choices are:

No filter.

First order low-pass,

$$G(s)=1/(s+1)$$

Second order low-pass,

$$G(s)=1/(s^2+2s+1)$$

Third order low-pass,

$$G(s)=1/(s^3+3s^2+3s+1)$$

Second order Butterworth,

$$G(s)=1/(s^2+2^{1/2}s+1)$$

Third order Butterworth,

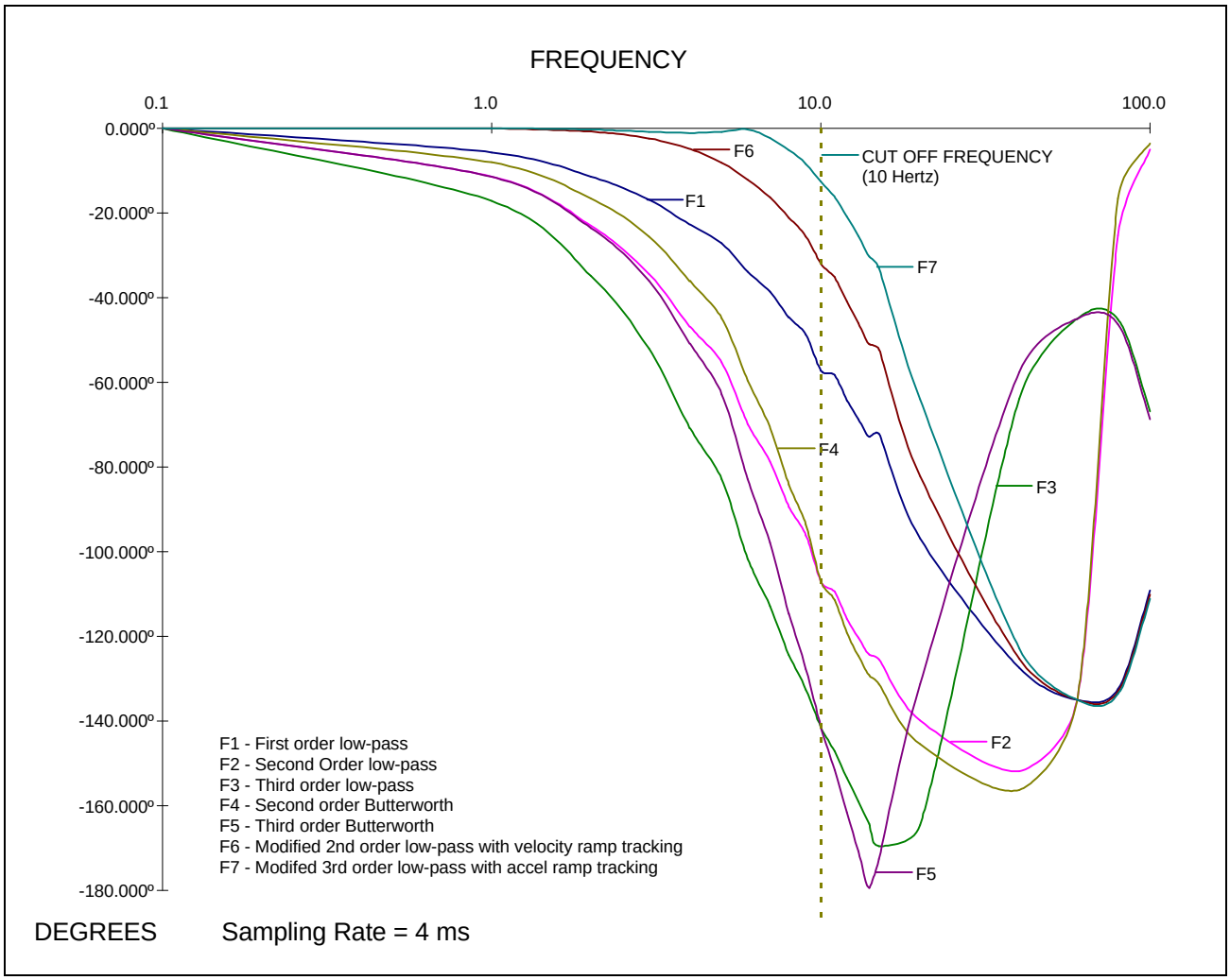
$$G(s)=1/(s^3+2s^2+2s+1)$$

Modified 2nd order low-pass with velocity ramp tracking,

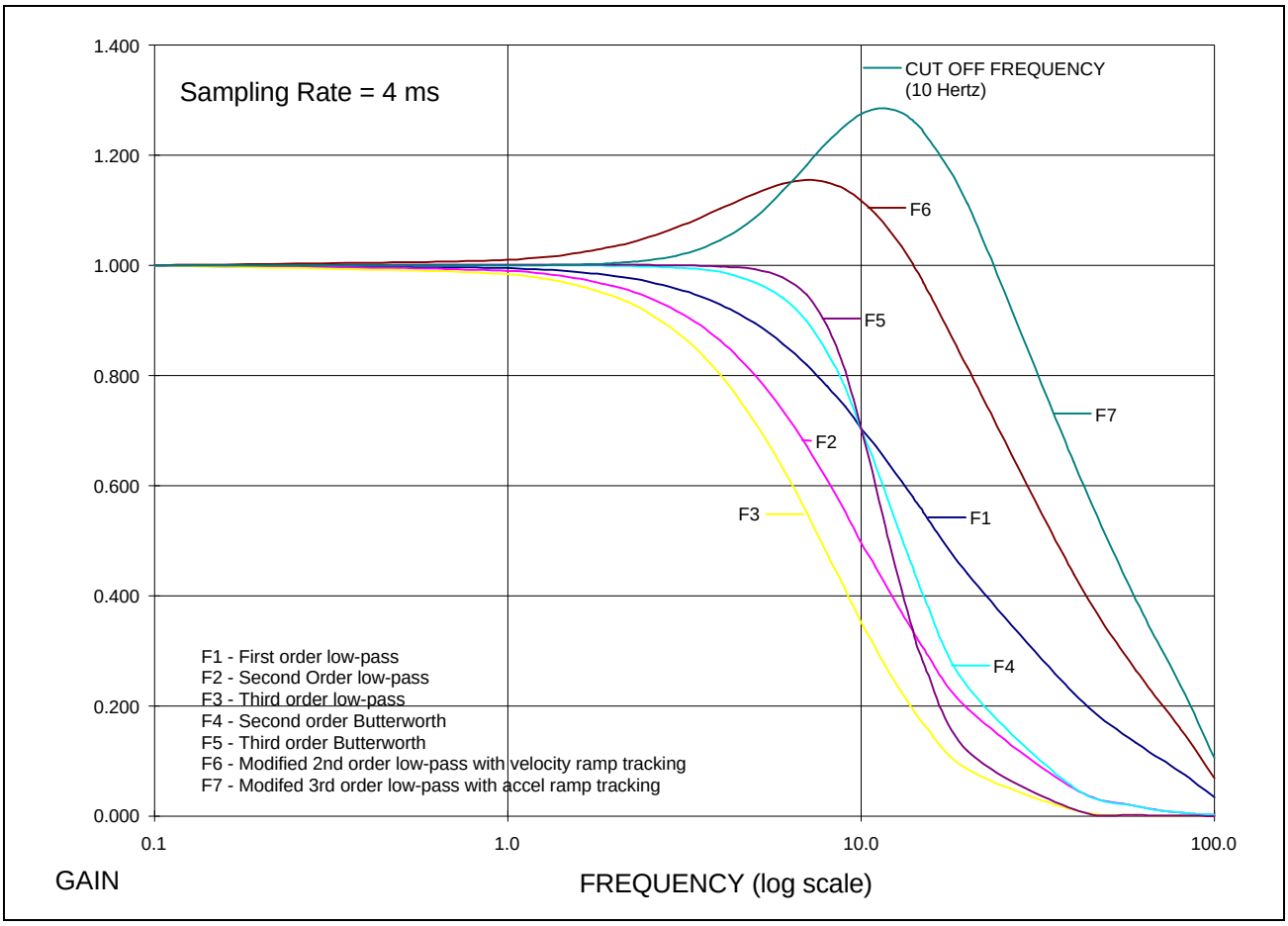
$$G(s)=(2s+1)/(s^2+2s+1)$$

Modified 3rd order low- pass with accel ramp tracking,

$$G(s)=(3s^2+3s+1)/(s^3+3s^2+3s+1)$$



The chart above shows the frequency vs. degrees for each filter. The cutoff frequency is 10 hertz and the sampling rate is 4 ms.



The chart above shows the gain vs. frequency for each filter. The cutoff frequency is 10 hertz and the sampling rate is 4 ms.

When a filter is chosen the cutoff frequency for the filter must be entered. For instance, the cutoff frequency for the First order low-pass filter is the frequency where the signal is reduced 3db [.707 gain, db=20*log(gain)].

The Properties screen shows the variable assignment used for this loop, these are read only.

More information on PID loops and Digital Filters can be found in the book “Computer Controlled Systems - Theory and Applications” by George Perdikaris, published by “KLUWER ACADEMIC PUBLISHERS”. 1991

Library of Congress Number: ISBN 0-7923-1422-0

PLS

Overview

This menu item spawns a API for viewing and editing the PLS tables on the CLC card and DDS2.2 version 4 drives. Each CLC program has a 16-position programmable limit switch(PLS), each drive a 8 position PLS. The card PLS can be driven by an external feedback or an internally generated position command(Virtual Master). The drive PLS is driven by it's primary or secondary axis feedback.

Menu Items:

File	*Programs:* Selects a program on the CLC card. Each program has a unique PLS. "Currently active" choses the active program. *Exit:* Exits this utility.
Edit	Display the edit dialog for the PLS chosen.
Selection	Choose a card or drive based PLS for display or edit.
Settings	Selects the DDE topic to communicate with a CLC card.
Refresh All	Warm boot of this utility, all program name, labels and current card PLS values are uploaded.

The initial screen is only for viewing, it graphically displays the "ON" period of each switch. On the bottom is the assigned register and current offset. To the right is the status of the assigned register bits and bit labels. For display purposes the default range of the graph is 0 to 360, other ranges can be entered on the edit screen.

To edit the PLS, select the EDIT menu item, or click on the graph on the switch to be set. In either case, another dialog box will appear, change the fields and press the SAVE button to send to CLC. Based on the type of PLS(card or drive), some of the the following PLS variables can edited on this screens:

Assigned Register

Master Type

Master Axis(card)

Current Offset(card)

Switches On Position

Switches Off Position

Lead Time(drives)

Graph Limit Minimum Value

Graph Limit Maximum Value

Card PLS Overview

The CLC provides a 16-position programmable limit switch per program. This limit switch can be based on either an external feedback or an internally generated position command. The PLS can be modified and set in a program utilizing the “Calc” icon or interactively through the PLS display. In the “Calc” icon, individual elements of the PLS can be changed (i.e. PLS[1].{a, o, r, t, on1-on16, off1-off16}. The PLS can be initialized, at compile time, using the PLS icon.

Limit Switch Operation

The programmable limit switch (PLS) is defined as 16 sets of position ranges for outputs. When the master position is in the On range, the outputs are on. When it is in the Off range, the outputs are off. The position ranges can overlap. The positions must be greater than or equal to the positions in the table to be in a position range. The position ranges in the table are defined clockwise, but the switch operates in either direction. If both the On range starting position and the Off range starting position are 0, the output is always off.

Example Programmable Limit Switch

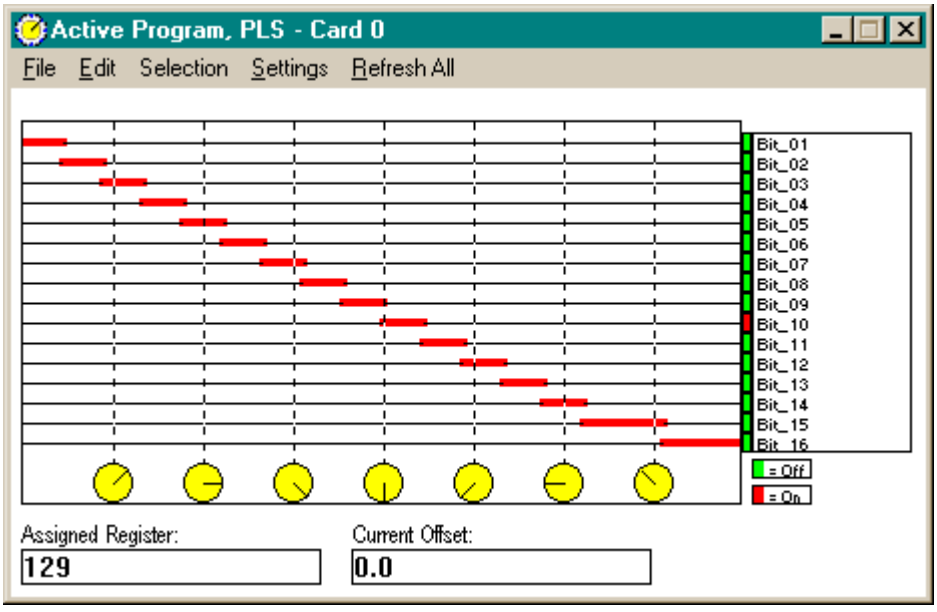
Output	On Position	Off Position
1	100	160
2	30	220
3	90	80
4	200	290
5	0	350
6	270	100
7	180	0
8	95	85
9	130	185
10	27	95
11	122	330
12	330	122
13	310	10
14	49	139
15	0	0
16	0	0

Output States at 90 degrees
(phase offset = 0)

Output	State
1	OFF
2	ON
3	ON
4	OFF
5	ON
6	ON
7	OFF
8	OFF
9	OFF
10	ON
11	OFF
12	ON
13	OFF
14	ON
15	OFF
16	OFF

The **On** range for Output 1 starts at 100 degrees, and the **Off** range at 160 degrees. The master is moving continuously. Output 1 will be on for 60 degrees between 100 and 160, and off for 300 degrees between 160 and 100.

The PLS menu item spawns an API for viewing and editing the PLS table on the CLC card.



A listing of programs on the CLC card is provided under the File menu item. “Currently Active” is also listed to access the PLS of the active program.

The initial screen is only for viewing, it graphically displays the “ON” period of each switch. On the bottom is the assigned register and current offset. To the right is the status of the assigned register bits and bit labels.

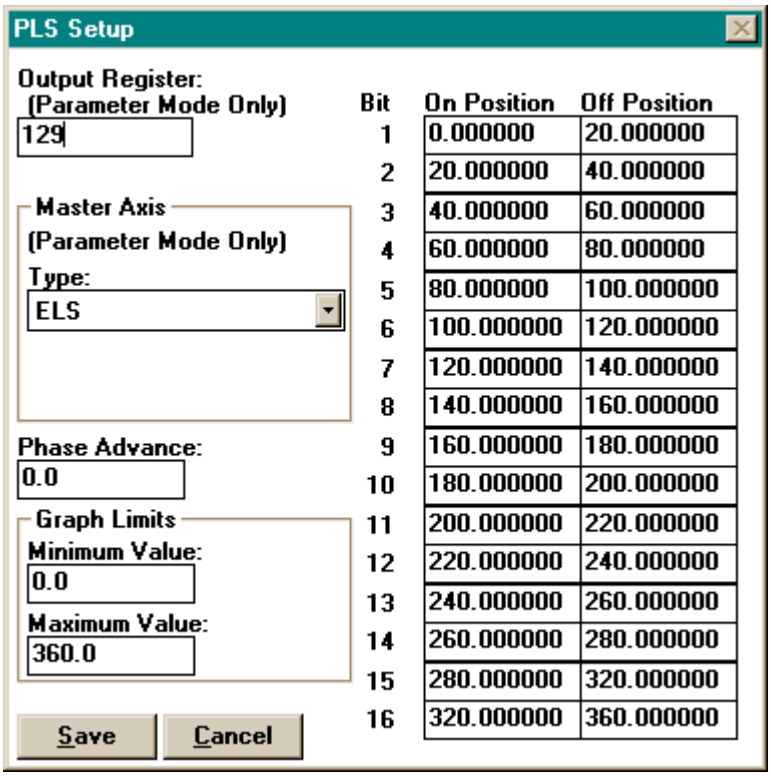
To edit the PLS, select the EDIT menu item, or click on the graph on the switch to be set. In either case, another dialog box will appear, change the fields and press the SAVE button to send to CLC.

The following PLS variables can be edited on this screen:

- Assigned Register
- Current Offset
- Switches On Position
- Switches Off Position

Phase Offset

The programmable limit switch has a phase advance which allows outputs to be turned on or off before or after their programmed position. The phase advance applies to all PLS outputs, and may be changed on the fly to correspond to different products or line speeds. The phase advance is specified as a position in degrees. This position is added to each position in the programmable limit switch table. For Output 1 in the example above with a phase advance of 5 degrees, the output will be on between 105 degrees and 165 degrees.



Master Type

The master axis is parameter-defined, with the following choices:

1. ELS master
2. Virtual master
3. Real master
4. On Drive

For many applications, the **ELS master** will be used. The ELS Master option allows synchronization of the limit switch outputs with the rest of the machine. The PLS will follow the same master that the ELS axes are following.

When the **Virtual Master** option is selected, the CLC always follows the ELS virtual master, even if the ELS slave axes are following a real master.

When the **Real Master** option is selected, the CLC follows one of the CLC's real masters. The real master can be independent of the ELS master, and includes a set of parameters to define gear ratio, feedback type, and

filtering. In the current CLC version, there can be up to two real masters in the system.

The **On Drive** option uses the PLS function built into Indramat DIAX03 type drives. As many PLS's as there are drives can then be defined in the system. The accuracy of the PLS is much better, since it is updated directly at the drive. The PLS is stored on the drive in parameters P-0-0131 to P-0-0135. The PLS_init command automatically sets these parameters when the program is first activated.

Register Designation

For maximum performance, the programmable limit switch is assigned to a register on the CLC. The I-O Mapper and image table are bypassed for this register. This allows drive-based I-O to be updated as the position is updated, regardless of the I-O scan time. This register can be associated with DDS DEA outputs or any other I-O card or PLC register. The register designation is defined in a system parameter.

The PLS can be disabled initially or "on-the-fly". If the assigned register number is set to 0, the PLS is disabled. The CLC leaves the outputs at the last state, and allows the user program and I-O Mapper to control them. If the application requires clearing the I-O after the PLS is disabled, the register must be cleared in the user program immediately after the PLS is disabled.

Drive Based PLS

Overview

The DIAX03 drives, version 04, include an 8-position limit switch that can be driven from the primary or secondary axis feedback. The drive includes parameter lists for on positions, off positions, and lead times. The status of the position switch is evaluated by the drive every 2ms, and an IDN in the drive is set accordingly. The outputs can be updated immediately by the drive, or in the next I-O cycle by the CLC.

Differences Between Drive and CLC PLS

The CLC-based PLS is stored with the user program as a separate table, and can be tied to any ELS master or real master. It has 16 switches, with a single position-based phase offset for all switches. Its elements can be easily changed at any time using a CALC expression.

The drive-based PLS is stored on the drive in parameter lists. It has 8 switches, with a separate time-based phase offset for each switch. The elements of this list cannot be changed individually. Changing one element of the list requires sending the entire list through the service channel.

PLS Type	Stored in	Outputs per PLS	Phase Offset	DEA outputs updated	Elements changed
CLC	user program on CLC	16	one position-based offset	by CLC 2ms after position	immediately
Drive	EEPROM on DIAX03 drive	8	individual time-based offsets	by drive at same time as position	via parameter list (up to 200ms per list)

User Program

Unlike the CLC-based PLS, the limit switch cannot be compiled into the program or changed dynamically via the CALC command. Therefore, the existing PLS/INIT command and the PLS table operands will apply only to CLC-based PLS. Drive based PLS are setup interactively using the PLS utility.

CALC icon

The CALC icon allows the drive PLS to be changed. The time to execute this command will vary greatly based on the usage of the service channel. The syntax is:

```
PLSD[1].on1 = 50        ; Set drive 1 PLS switch 1 on at 50
PLSD[2].off4 = 200      ; Set drive 2 PLS switch 4 off at 200
PLSD[1].lt1 = 0         ; Set drive 1 PLS switch 1 lead time to  50
```

Assigned Register

Changes to the assigned register are saved to the drive but do not take effect until the card makes the transition from “Parameter Mode”. ***See A-0-0009***.

Points

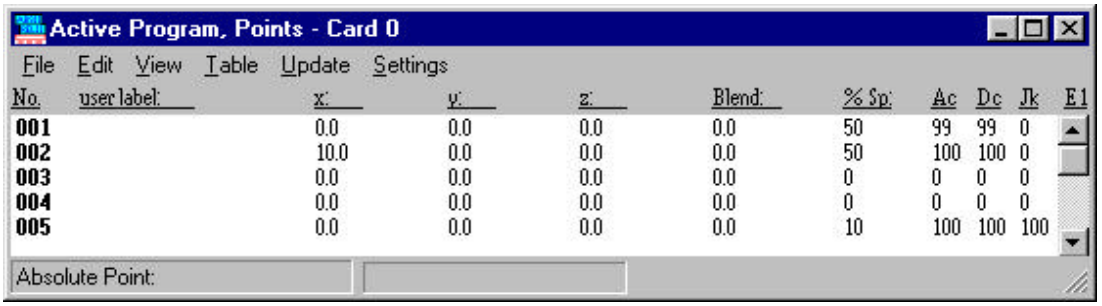
The View Point Table command permits the user to view and edit the absolute and relative points in a program that has already been downloaded to the CLC card. The dialog box automatically uploads and displays the contents of the point table for the CLC resident program that was last selected in the **File/Program Select** command. If no program has been previously selected, VisualMotion defaults to the active program.

Points are used in **coordinated** motion programs to describe a location in Cartesian coordinates, tool orientation, and associated events.

The point table can also be referred to as a display of raw information for building cams using the VisualMotion CamBuild Icon. For more information***, refer to CamBuild Icon*** in the chapter "Icon Programming."

To use points, you must allocate memory in the VisualMotion ***Size Icon***.

Each point has: { x, y, z, blend, speed, acceleration, deceleration, jerk event 1, event 2, event 3, event 4, roll, pitch, yaw, and elbow } fields.

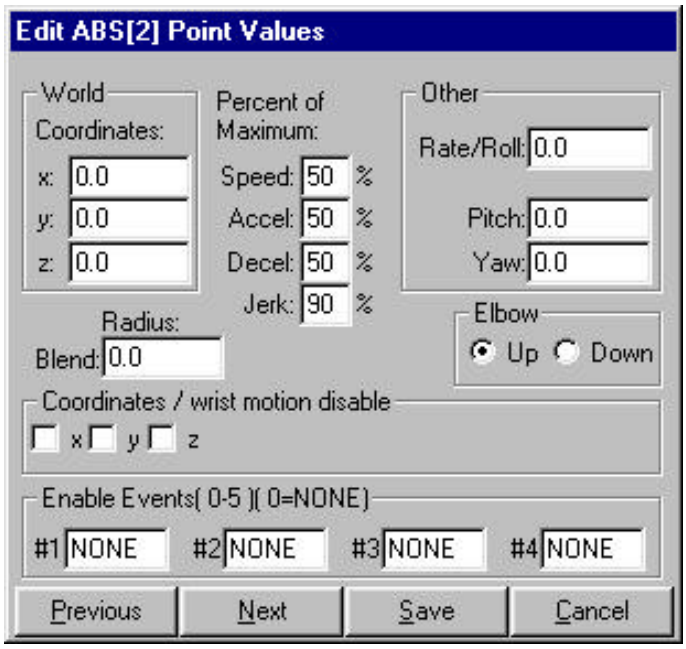

Active Program, Points - Card 0
File Edit View Table Update Settings
No. user label: x: y: z: Blend: % Sp: Ac Dc Jk E1
001 0.0 0.0 0.0 0.0 50 99 99 0
002 10.0 0.0 0.0 0.0 50 100 100 0
003 0.0 0.0 0.0 0.0 0 0 0 0
004 0.0 0.0 0.0 0.0 0 0 0 0
005 0.0 0.0 0.0 0.0 10 100 100 100
Absolute Point:

To better navigate long tables, the user may click and drag the vertical scroll bar button while looking at the point number indicator that appears at the right of the dialog box title bar. The point number corresponds to the top of the list when the scroll bar button is finally released. The user must expand the viewing window to view the additional roll, pitch, yaw and elbow properties used with a six-axis CLC system.

Another viewing aid is the **View** menu, which presents a list of the point elements and allows the user to deselect any number of them in order to exclude them from the display.

The **Table** menu allows the user to switch between the absolute and relative point tables.

Points may be edited by double-clicking on a point in the list, or highlighting the point and then clicking **Edit**. Editing a point table entry opens an absolute or relative Edit Point Values dialog box, depending on which type was selected with the Table menu.



The Edit Point Values dialog boxes permit individually changing the values for each point table entry on the CLC card. Clicking on Save immediately downloads the new values to the program on the CLC card. The changed values are displayed at the next automatic update.

Note: *Coordinated moves using an ABS/REL table point with zeroes in any of the fields for Speed, Acceleration, Deceleration, and Jerk will default to 1%.*

The first time a point is taught and the point has zeroes in the Speed, Acceleration, Deceleration, and Jerk fields default values are loaded. The defaults are 10% for speed, 100% for acceleration, 100% for deceleration and 100% for jerk.

For more information on the absolute or relative point tables and how they are used, ***refer to the appropriate section in the chapter, "Programming Concepts".***

In the Coordinates/wrist motion disable section of this screen, the user can disable coordinated motion on one or more axes for a particular point. To include this feature in the VisualMotion program, refer to the section ***Calc Icon***in the chapter, Icon Programming.

In the Elbow Properties section of this screen, the user can choose the Up or Down radio button to determine the elbow direction for a particular point. To include this feature in the VisualMotion program, refer to the section ***Calc Icon*** in the chapter, Icon Programming.

Registers

This Windows utility allows viewing and editing of registers on the CLC card. There are 1024 registers on the CLC-V card, and 512 registers on the CLC-P and CLC-D. ***See the Control and Status Registers*** for a description of the registers reserved by the CLC for system control and status.

Registers can be changed by editing the register, editing the register's bits, or forcing the register bits. Register priority is as follows from highest to lowest:

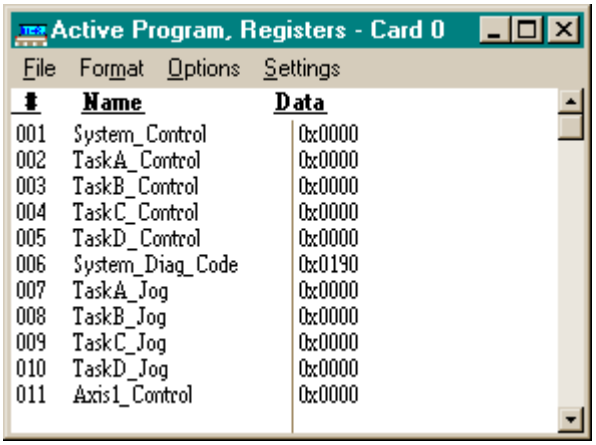
- VME/PC Access - registers written directly by an external device.
- I/O Drivers - registers control by I/O device drivers.
- Register Forcing - via serial port.
- I/O Mapper - via I/O mapper equations.
- Direct Register Access - via serial port or program instructions.

On the CLC/V these registers are accessible from the VME address space by other VME cards. On the CLC/P these registers can be accessed through the twoport RAM on the ISA bus by the PC.

The Physical Name of the registers in the listbox are taken from register labels, register labels can be assigned to non-system registers.

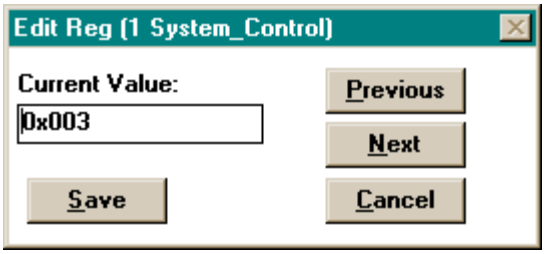
Data Display - These radio buttons selects the display format of the register data.

Choosing the Registers menu item from the Data menu opens a Registers dialog box listing the register ID numbers and the symbolic label names associated with the registers (from the register.lst file). When the dialog box opens, it automatically uploads and displays the current contents of the CLC registers. Note that the CLC has a maximum of 1024 registers.

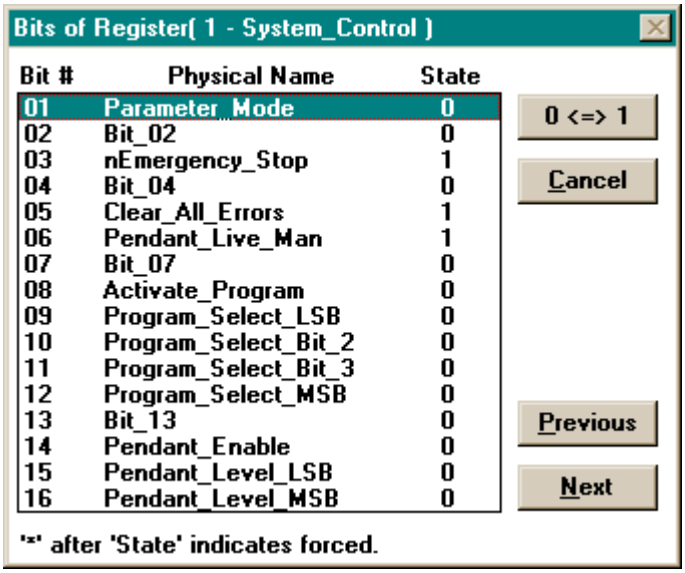


The **Format** pull-down menu allows you to change the displayed format of the register data.

The **Options** pull-down menu shows the available F-key commands for editing, forcing, and clearing. Selecting a list entry and pressing F3 opens an Edit Reg dialog box allowing change of the value in the selected I/O register table. Clicking the Save button downloads the changed value to the CLC.



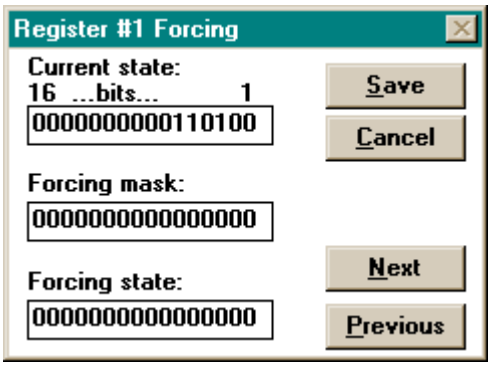
Selecting a list entry and pressing F2, or simply double clicking on a list entry, opens a Bit Names dialog box for the selected register and displays the bit names and values associated with the register. Bit names are obtained from the "bit.lst" file in the Host system :\CLC sub-directory and are not uploaded from the CLC.



Selecting a list item and clicking on the "0 <=> 1" button inverts the current state of the bit and downloads the change of state to the CLC through the serial communication link. The next periodic upload from the CLC reflects the change of state in the displayed list.

Bits controlled by the CLC I/O, user program, or I/O Mapper will not be altered using the "0 <=> 1" toggle button. However, their state may be "forced" using the Forcing option (F4). Items denoted with an "x" in the rightmost column have forcing in effect for the associated bit.

System installation and troubleshooting may require directly changing the state of register bits without depending on the I/O sub-system. Selecting a register and pressing F4 opens a Register Forcing dialog box that allows you to setup a forced bit during system setup and debugging.



Forced bits cannot be affected by a CLC program or the I/O sub-system. Forcing directly accesses the CLC system's I/O lines. Bits that are forced will remain in the forced state until the forcing is changed, the CLC card is reset, or power is cycled on-to-off.

Forcing employs forcing mask and forcing state 16-bit control words allowing you to change a single bit, or combination of bits, within the register without affecting the other bits.

The Forcing mask value enables which bits may be affected by forcing. The Forcing state value determines the actual state of the enabled bits. The following algorithm mathematically describes how the final register state is set:

New register state = (old state & inverted forcing mask) | (forcing state & forcing mask)

Example:

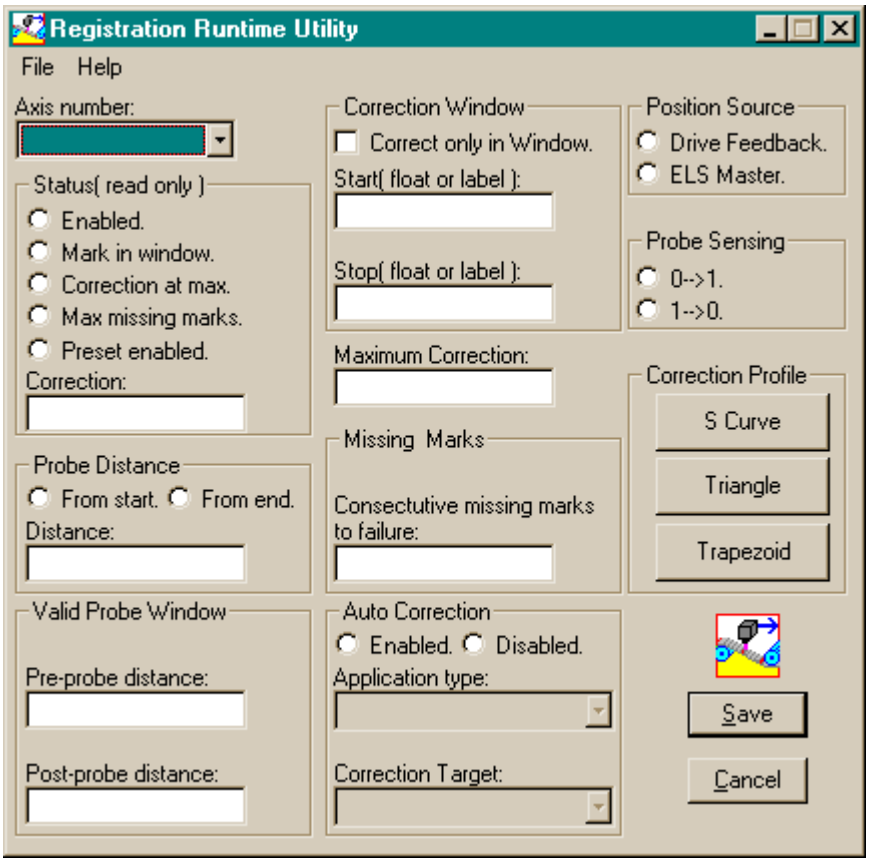
```
forcing mask           0000 0000 0000 1000
forcing state          0000 0000 0000 1000
inverted forcing mask  1111 1111 1111 0111
old register state     0000 1101 1010 0001

new register state     0000 1101 1010 1001
```

WARNING: *Bit forcing directly changes the state of the CLC's inputs and outputs. Forcing I/O bits can result in harm to people and equipment. Make sure you fully understand all the effects on the system that could result from forcing an I/O line.*

Registration (Not Functional for GPS 6.0)

The CLC Registration Runtime Utility allows users to access the Registration Icon configuration settings from the VisualMotion Menu. ***See Chapter 6 - Registration Icon*** for more information.



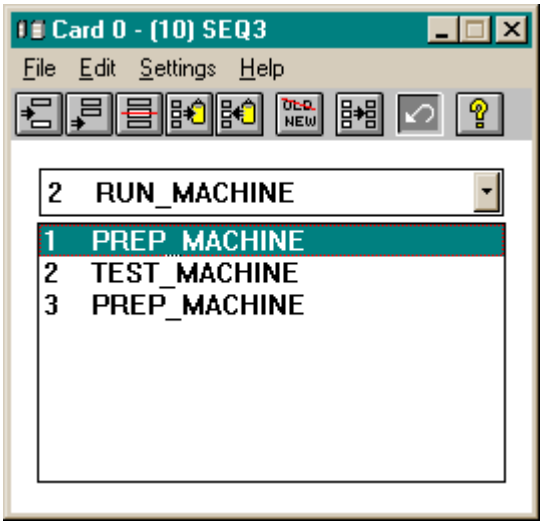
Sequencer

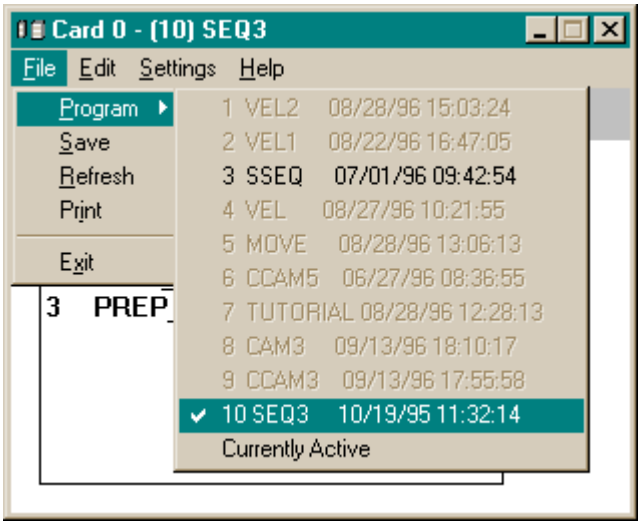
The CLC Sequencer provides the user with a facility for making machine operational changes without having to edit, recompile and download a program to the CLC Card. A Sequencer is a list of steps which contain one or more functions, with up to five arguments per function.

When the Sequencer is selected from the Data menu, Visual Motion reads the Sequencer information from the CLC card and displays it in the Sequencer editor. This allows the user to view and edit a Sequencer list on-line. An error will occur if there are no Sequencer lists in the active program. A Sequencer List needs to be initialized using the Sequencer icon before it can be edited on-line. ***See Chapter 5 - Programming Concepts*** for more information.

Sequencer Editor

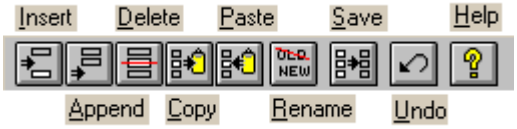
The caption on the window identifies the active CLC card number and program. The initial screen provides a drop down *combo-box* which lists all the Sequencer list names found in the selected program. The box below it shows the order of all the Steps used with the selected Sequencer.

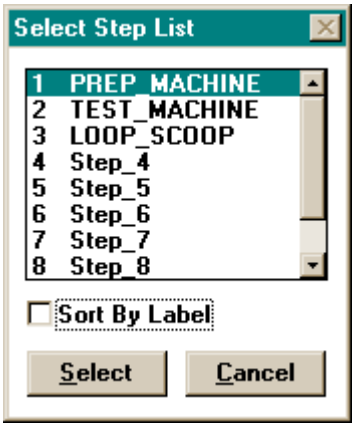




Select “Program” under the “**File**” menu to list programs currently on the CLC card. When a different program is selected, the associated Sequencer tables will be uploaded and the program name will appear in the window caption. The “File” menu also allows the user to save refresh or print the Sequencer table.

The “**Edit**” menu items allow the user to insert, append, delete, copy, paste and rename steps within a Sequencer list. This editing is also available through icons on the tool bar.

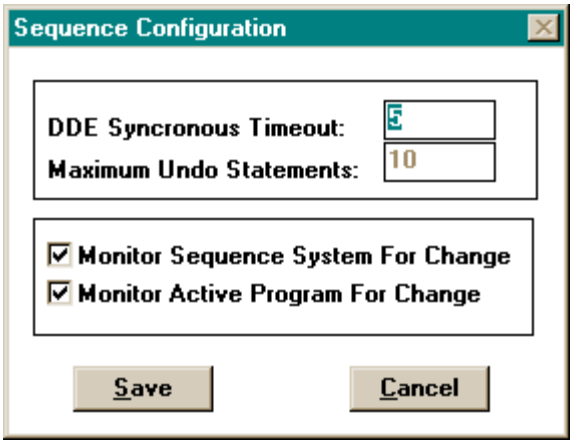




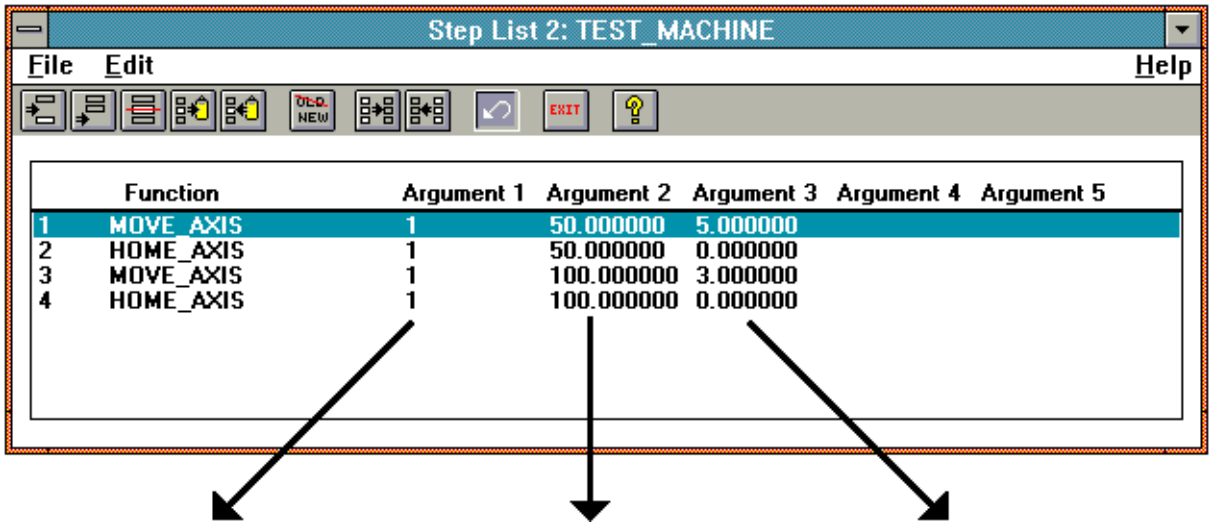
The steps can also be moved by selecting and dragging them with the mouse. Select Step List to show all available Steps.

The “Connection” item under the “Settings” menu allows the user to select one of the following connections: serial, XYCOM VME, PC ISA bus, AT Modem, Demo, GE Plug & Play or CLC File. The card number, 0 through 15, can also be selected. The DDE server name is also listed along with it's topic string. This information is useful when creating client applications which can *talk* to the CLC communications server. ***See section Card Selection***.

Selecting "Configuration" allows the user to adjust the DDE Synchronous Time-out. This is the amount of time in seconds the Sequencer Editor will wait for a response from the DDE server before proceeding. **See *Appendix A - CLC DDE Server* for more information**. Sequencer Configuration also allows the user to monitor the Sequence System and/or Active Program for change.



Functions and arguments can be edited by double clicking on a selected step. A Step List window will open with the same edit and icon tools on top. Individual arguments can be edited by double clicking on them. After making any changes the Step List needs to be saved to retain the current edits in the Sequencer.

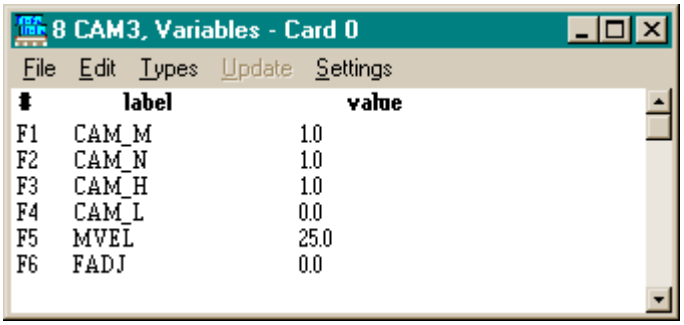

Step List 2: TEST_MACHINE
File Edit Help
Function Argument 1 Argument 2 Argument 3 Argument 4 Argument 5
1 MOVE_AXIS 1 50.000000 5.000000
2 HOME_AXIS 1 50.000000 0.000000
3 MOVE_AXIS 1 100.000000 3.000000
4 HOME_AXIS 1 100.000000 0.000000

Argument value adjustments can also be sent directly to the CLC card while the program is running by selecting Send. This can only be done if the Step List order hasn't changed since the last time it was saved. If a change has been made to the function order, this option will gray out. A Step List can not be saved to a Sequencer if it is running.

An entire function can be edited or replaced by double clicking on the function name. A list of available functions will open. These are all the predefined subroutines within the Visual Motion program. After selecting a function, the user will have to enter a value for each argument. This value will have to be within the programs predefined limits. Selecting a function this way changes the order of the Step List. The user will not be able to *send* the resulting argument values to the CLC card without first saving the Step List.

Variables

The CLC Variables dialog box permits viewing the integer and floating point variables of a CLC resident program. The dialog box automatically uploads and displays variables of the active program on the CLC. The list display and variable selection for editing is similar to the View Point Table dialog box. The program is selected with File/Program Select and the variable type is selected under the Types menu.



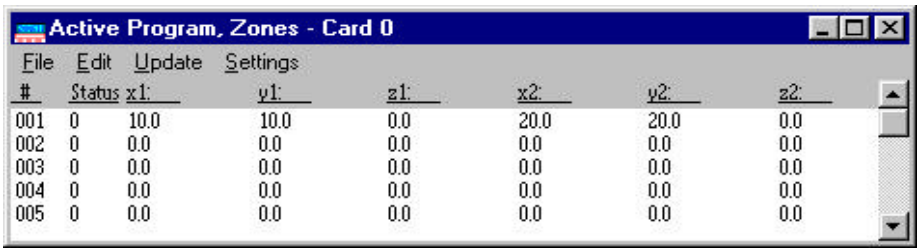
Selecting a list entry and clicking the Edit button, or double clicking the list entry, opens a Variable Edit dialog box allowing change of the value in the selected variable in the CLC resident program. Exiting the dialog box automatically downloads the changed values to the CLC.

Zones

This Windows utility allows viewing and editing of the zone table on the CLC card. Zones can be used in coordinated motion programs to describe a volume of space where motion of any kind is prevented.

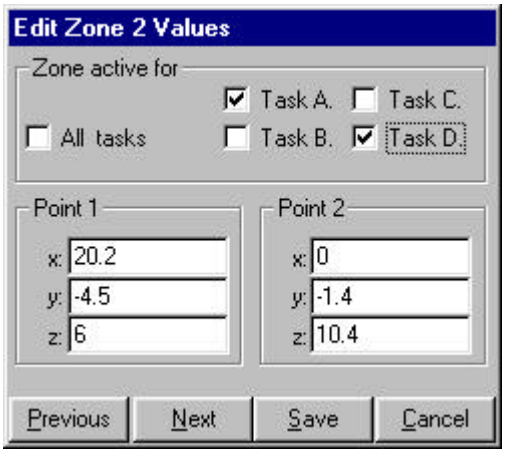
Programs on CLC card - program selection is through a menu selection containing a list of programs on the CLC card plus "Currently active."

To edit a event, select it in the list box, and press the EDIT button, or double-click on the list item. In either case, another dialog box will appear with a edit field. Change the fields and press the SAVE button to send to CLC.



Each zone is defined by the { x, y, z } coordinates of two opposite corners (Point 1 and Point 2) of a cube in space.

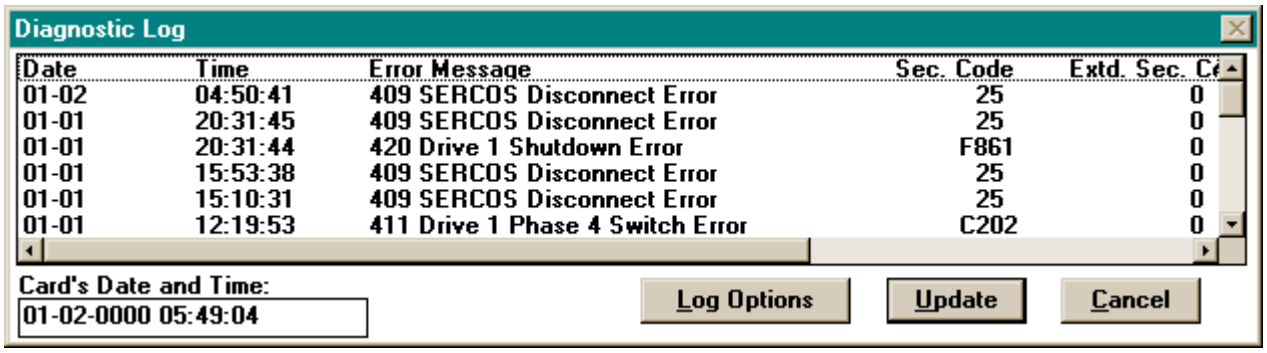
For each defined zone, the status can be ACTIVE (checked) or INACTIVE (unchecked) for selected tasks, no task or all tasks. For example, the zone defined in the “Edit Zone 2 Values” screen below is active for zones A and D.



4.8 The Status Menu

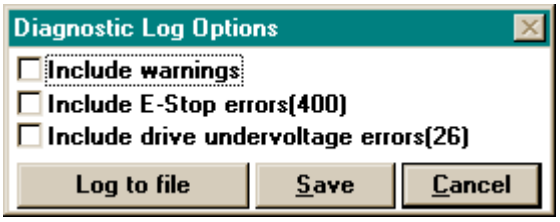
Diagnostic Log

This window lists the last 100 errors the CLC card has encountered. Along with the error messages, the date, time and extended error codes are displayed.


Diagnostic Log
Date
Time
Error Message
Sec. Code
Extd. Sec. Cd
01-02 04:50:41 409 SERCOS Disconnect Error 25 0
01-01 20:31:45 409 SERCOS Disconnect Error 25 0
01-01 20:31:44 420 Drive 1 Shutdown Error F861 0
01-01 15:53:38 409 SERCOS Disconnect Error 25 0
01-01 15:10:31 409 SERCOS Disconnect Error 25 0
01-01 12:19:53 411 Drive 1 Phase 4 Switch Error C202 0
Card's Date and Time:
01-02-0000 05:49:04
Log Options
Update
Cancel

Date and time are relative to the power on of the CLC D/P cards, they have no battery backed clock. The time can be set in card parameter ***C-0-0126***.

The Log Options button opens an options window. It allows some common errors to be ignored and saving the diagnostic log to a file. Also see card parameter ***C-0-2020***

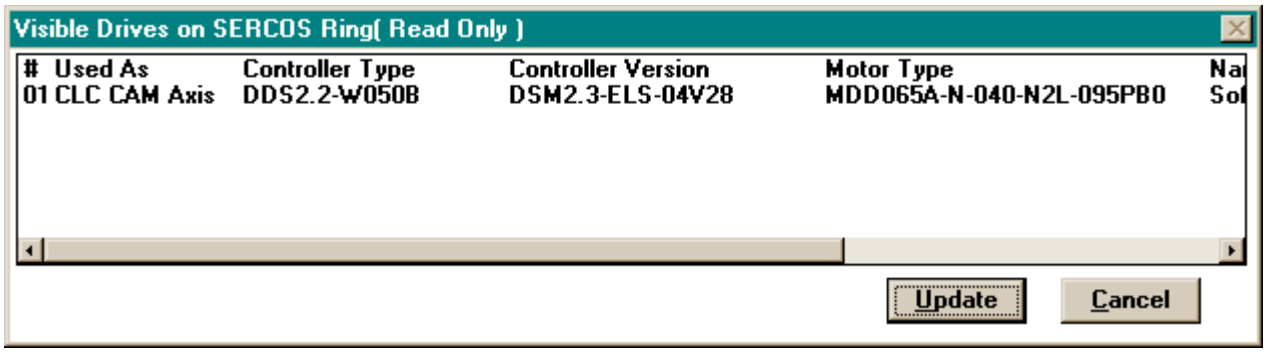


Drives

Refer to Appendix D. CLC Drive Parameter Editor.

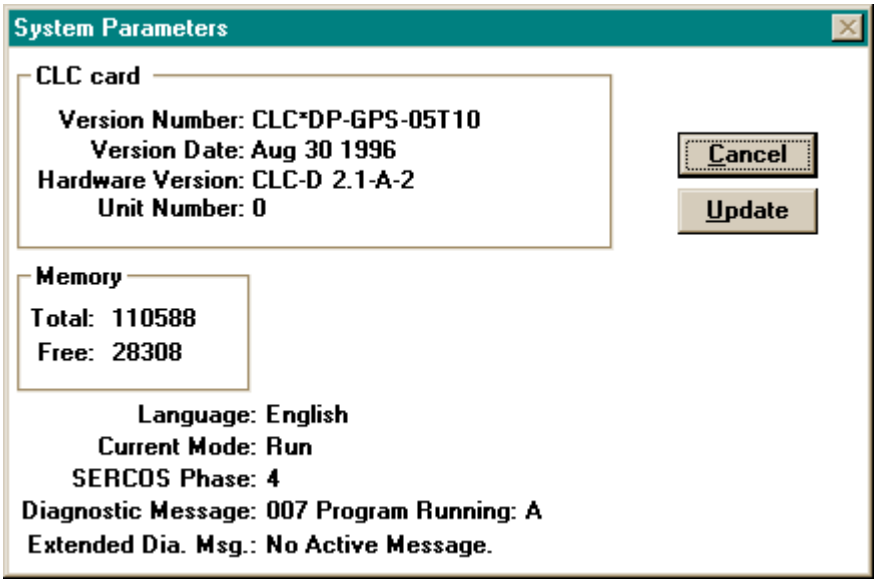
Drives on Ring

This window displays the addresses, uses and identification of the drives and motors the CLC card sees on the SERCOS ring.



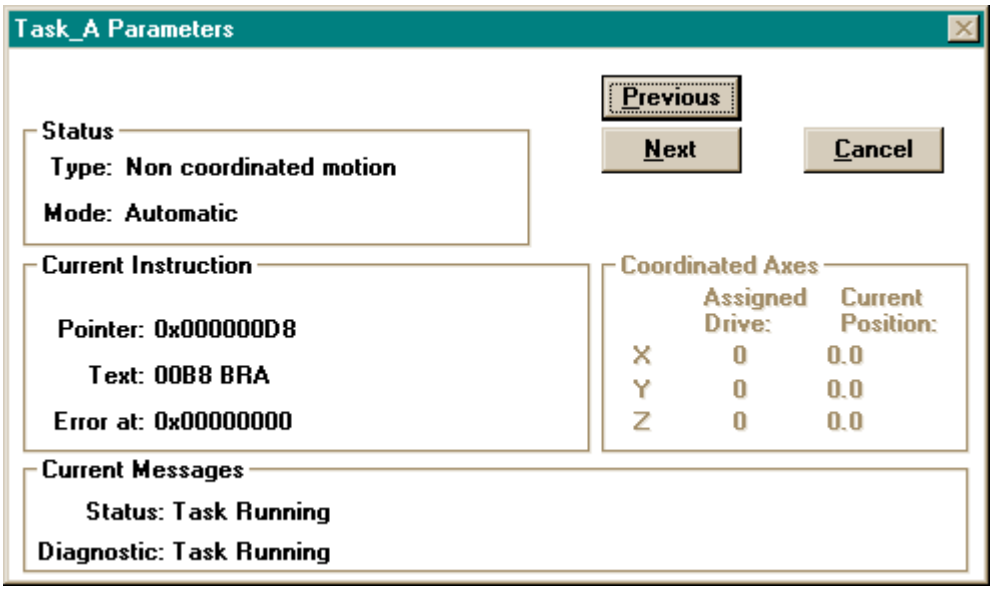
System

Choosing System from the Status menu opens the System Parameters dialog box. System Parameters displays information about the current CLC card hardware and software for the indicated unit number; and the total memory and free memory on the CLC card.



Tasks

Selecting Task from the Status menu opens the Task_A Parameters dialog box and uploads data regarding the current CLC task. The other tasks may be viewed by clicking on the Previous or Next buttons.



Status indicates the type of motion programmed in the selected task for the active program and the current CLC mode (Parameter, Initialization, Manual or Automatic).

Current Instruction displays the instruction executing and its pointer, and a pointer to a run-time error if one has occurred. This display is useful when debugging in single-step mode. If a program is running in automatic mode, the displayed instruction is the instruction that was executing at the time that the SERCOS cycle sampled instruction execution, which may appear to be random.

Current Messages displays the last messages encountered in the program.

Coordinated Axes displays the axes in the active task that are assigned to coordinated motion and their current position.

4.9 The Options Menu

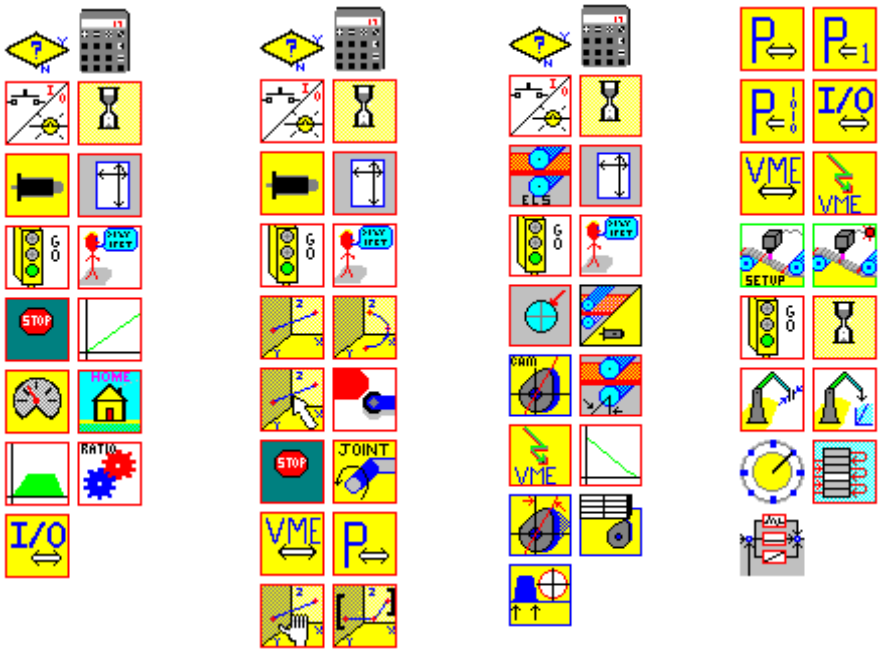
The Options menu provides pre-configured palettes for single axis, coordinated, and Electronic Line Shaft (ELS). Choosing one of these menu items loads the selection onto the palette area on the left of the VisualMotion workspace.

Icon Palette 'Single' from the Options menu selects a set of icons used frequently in single axis control.

Icon Palette 'Coord' selects a set of icons used frequently in coordinated control.

Icon Palette 'ELS' selects a set of icons used frequently in electronic line shaft(ELS) control.

Icon Palette 'Utility' selects a set of general purpose icons used in parameter setup.



Single **Coord** **ELS** **Utility**

Choosing **Icon Comments** enables/disables identifying comments that appear when the mouse cursor is moved over the top of the icon. The icon comments in a program flow can be modified in the corresponding icon setup box. The setup box appears when the icon is first placed or by double clicking the left mouse button while the pointer is over the icon. Each box has a "Label" button to edit its comment text, by default there is no comment. The comment also appears on the printout of the setup information, if enabled or not.

Choosing **Icon Labels** from the Options menu alternately turns the icon labels in the VisualMotion workspace on or off. If there are no user entered labels, VisualMotion uses the default icon labels.

4.10 Help

VisualMotion's Help menu accesses the on-line help system.

Selecting **Help/Getting Started** or pressing **<F1>** opens the main help screen, from which you can browse through the system by pointing the cursor at the green keywords (help topics) and clicking.

Selecting **Help/Search** opens a dialog box into which you can type a keyword to go directly to a specific help topic.

Help/Change Log displays a list of the new features that appear in your version of VisualMotion.

5 Programming Concepts

5.1 Overview

Indramat provides two methods for writing CLC motion control programs, the CLC Text Language and CLC Icon Language. Both methods of programming have many characteristics in common. This chapter describes these common elements and their implementation.

5.2 Program Tasks

The CLC provides up to four user-defined motion control tasks for control of up to forty axes. The CLC motion control tasks are named A, B, C and D. A single CLC task may support any number of non-coordinated axes. However, each task is limited to two or three coordinated axes of motion.

Command Execution

Normal multi-tasking is achieved by consecutively executing one instruction from each task. During normal operation, the CLC does not wait for an instruction to be completed before processing the next instruction. Also, the CLC's multi-tasking path planner and the use of Indramat DDS intelligent digital drives permits simultaneous execution of multiple motion commands.

The CLC path planner also uses several levels of priority to control the execution of program statements. For example, coordinated multi-axis motion may require some instructions to wait for a specific type of motion to complete before proceeding to the next instruction. The CLC priorities associated with user programs are listed below. A higher numbered task has priority over any lower numbered task.

Highest Priority 7 - CLC Executive Path Planner and Single Axis Events
Priority 6 - Event from Task A
Priority 5 - Event from Task B
Priority 4 - Event from Task C
Priority 3 - Event from Task D
Priority 2 - CLC Executive Timer Task (Repeating Events)
Lowest Priority 1 - User Motion Tasks A, B, C, D, Teach Pendant

5.3 Events

CLC events are a privileged form of subroutines. During the execution of an event, all CLC program tasks are suspended until all pending events have completed. The CLC system executive program maintains an event queue or stack for events associated with each task. This permits each task to have up to four pending events active at any time. See Chapter 4. ***Visual Motion Menu Commands*** for more information.

Events are prioritized according to Task, from A to D. Events associated with Task A will interrupt execution of events associated with Tasks B, C, & D. Events associated with Task B will interrupt events associated with Tasks C & D, etc.

To add an event subroutine to a user program, select “Add Event Function” from the “Edit” menu in VisualMotion. After naming the event function (subroutine), a blank page will appear for programming the event. Event functions must begin with a “Start” icon and end with a “Finish” icon, just like a normal subroutine. Any VisualMotion icon that executes during run-time can be used in the event function subroutine. Events are armed in the user program with various icons (see following sections), depending upon how the event is to be called

See Event Table for information regarding Event Table Data.

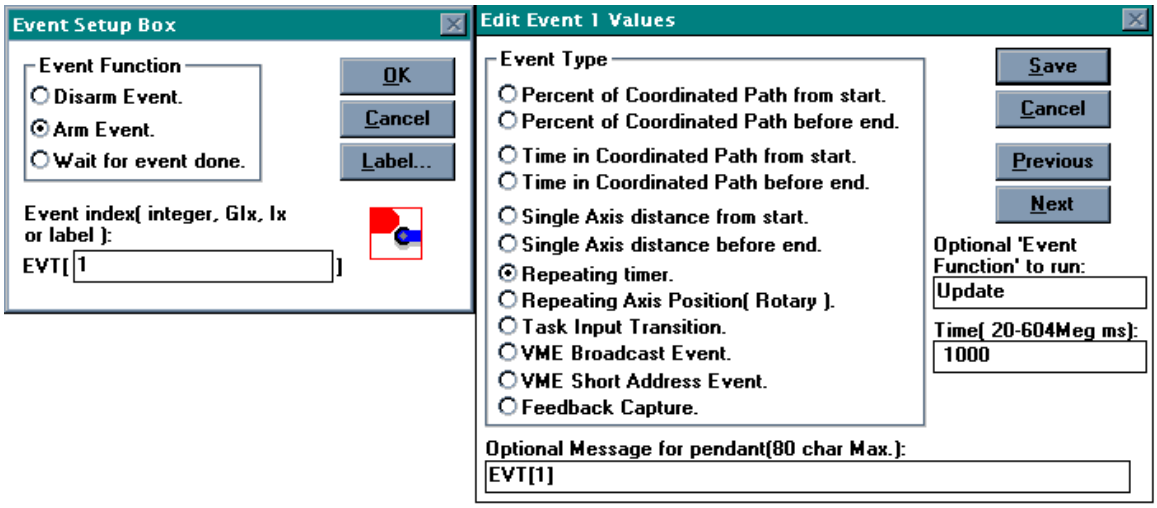
NOTE: *Because an event preempts all user tasks, events can adversely effect task response time and execution. Events should be used only when real-time response is needed and should be kept as short as possible. For example, involved calculations or waiting for other I/O will block the execution of all tasks that are associated with that event. If possible, use the event routine to set a flag value in an integer variable and immediately return. Then test the flag during an idle period and perform the required operations at the later time (don't forget to reset the flag).*

Time-based Events

The CLC provides two types of time based events, repeating timer events and events that are time related to a coordinated motion path.

Once a repeating timer event is armed in the user program, it executes periodically at the programmed time interval until the event is disarmed. The timer period for the event is specified in the event table (argument) and has a resolution of 20 milliseconds. The event number to execute every timer period is specified in the “Event” icon. After the user program is compiled and downloaded, the event type – Repeating Timer -- , event function, and timer period must be specified in the event table. The event table is loaded by selecting “Events” under the “Data” menu in VisualMotion.

Coordinated motion can provide time based events that are related to travel time along a specified geometry segment and are initiated by the path planner. These events execute at a fixed time period after motion starts or before motion ends on the specified segment. The time period is specified in milliseconds in the event table.



NOTE: *CLC single axis non-coordinated motion instructions use the internal intelligence of the DDS drives for positioning. Because the rate profile for single axis motion is developed within the drive, the time method of triggering an event related to motion cannot be supported.*

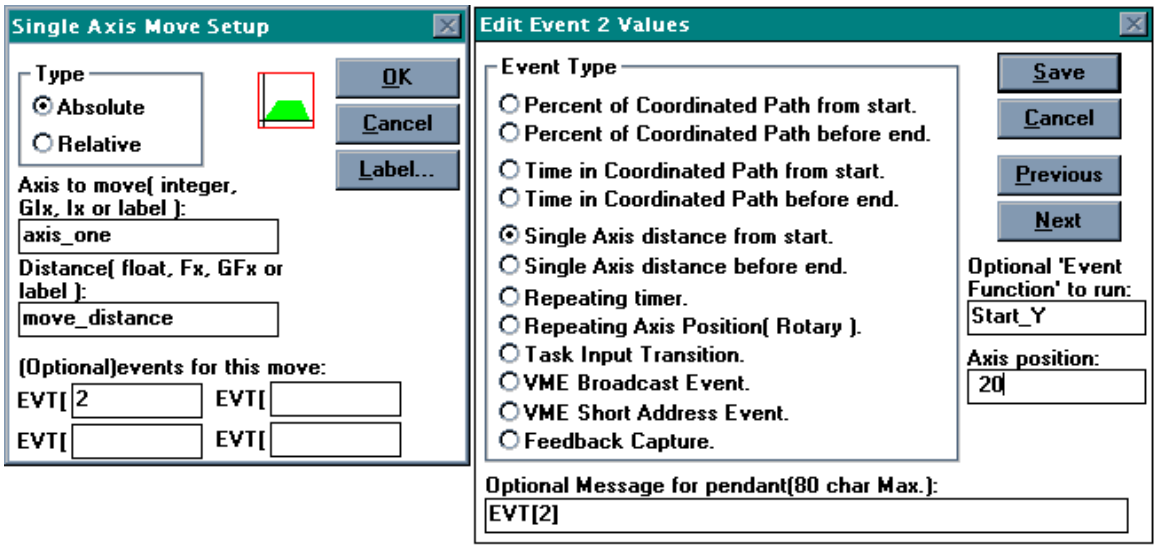
Distance-based Events

Distance based events are triggered after the distance specified has been traversed on the geometry segment. They can be triggered off both coordinated and non coordinated (single-axis) moves. The response time for coordinated motion events is dependent upon the number of coordinated axes that are programmed and the task that calls the event (task priority A - D). Each task using coordinated axes requires one additional SERCOS cycle. Therefore, the response time may vary from one to four SERCOS cycles.

Single axis non-coordinated motion requires specification of the trigger distance from the endpoint or beginning of motion using an absolute number of axis units. The event number to be executed is specified in the "Optional Events" section of the single axis move icon. After the user program is compiled and downloaded to the CLC the following information must be in the event table:

- Event type
- Single axis distance from the end *or* from the beginning
- Event function
- Trigger distance

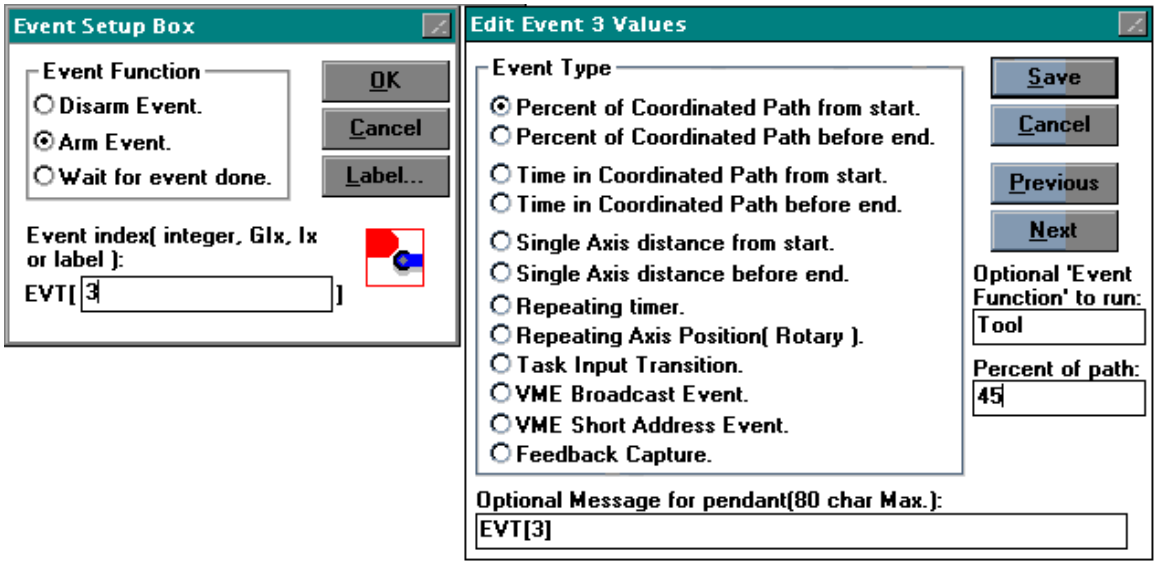
The event table is loaded by selecting "Events" under the "Data" menu in VisualMotion. Single axis distance based events cannot be used on the same axis with repeating axis position events.



Multi-axis coordinated motion requires the trigger distance specified as a percentage of the total length of the segment. The event number to be executed is specified in the "Event" icon on the "Coord" icon palette. After the user program is compiled and downloaded to the CLC, the following information must be specified in the event table:

- Event type
- Percent of coordinated path from start or before end
- Event function
- Trigger distance

The event table is loaded by selecting "Events" under the "Data" menu in VisualMotion.



When specifying a distance-based event trigger with coordinated motion, an event occurring within a blend segment may not trigger as anticipated. The range of potential paths that could be generated by the path planner through the blend segment must be considered. Consider when one segment is blended into another, with one event set to trigger near the end of the first segment and another event set to trigger near the beginning of the next segment. If the blend radius specified is sufficiently large, the second segment may blend into the first segment far enough that the second segment event triggers before the first segment event can occur. This situation can be controlled by ensuring that the programmed blend radius is smaller than the specified trigger distance of the second segment.

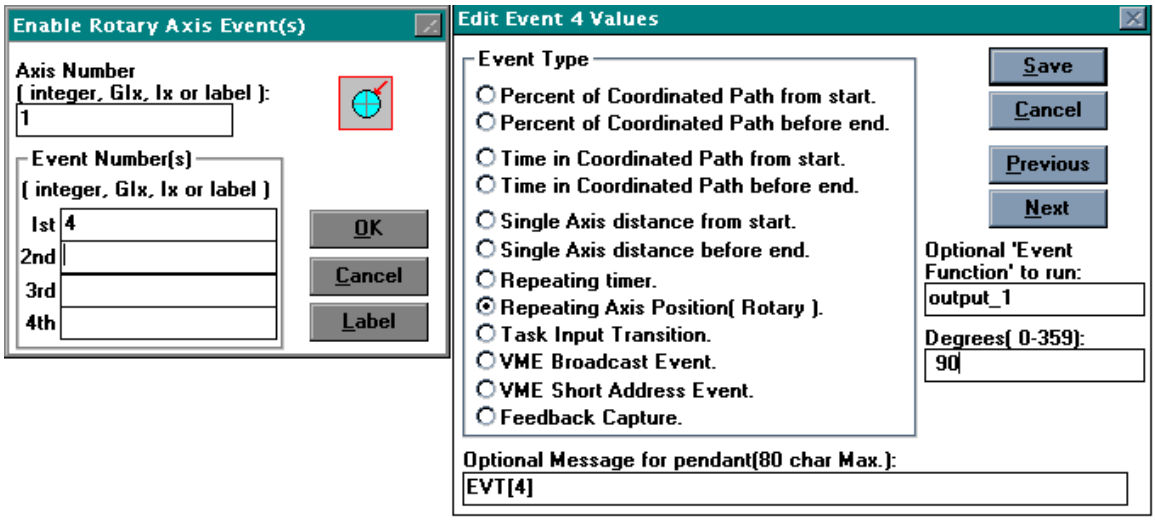
Repeating Axis Position (Rotary) Events

Repeating Rotary Events trigger each time an axis encounters an absolute position. The axis can be in single-axis, ELS, ratio, or velocity mode and can be configured for modulo or non-modulo positioning. ***See AxisEvt Icon description***.

Each time the event position is passed in either direction, the function specified in the event table is executed. Since rotary motion uses the shortest path to reach the next specified position, you should take care that the axis will actually travel through the position specified to trigger the rotary event. Otherwise, the event will not occur.

Repeating rotary events only need to be armed once in a user program. After the user program is compiled and downloaded to the CLC, the event type – Repeating Axis Position, event function and trigger position must be specified in the event table for each rotary event. The event table is loaded by selecting “Events” under the “Data” menu in VisualMotion. If changes are made to the arguments of rotary events in the event table, then the “Rotary Events” icon must be executed again in the user program before the changes will take effect.

NOTE: *Rotary events cannot be attached to an axis with single axis distance based events.*

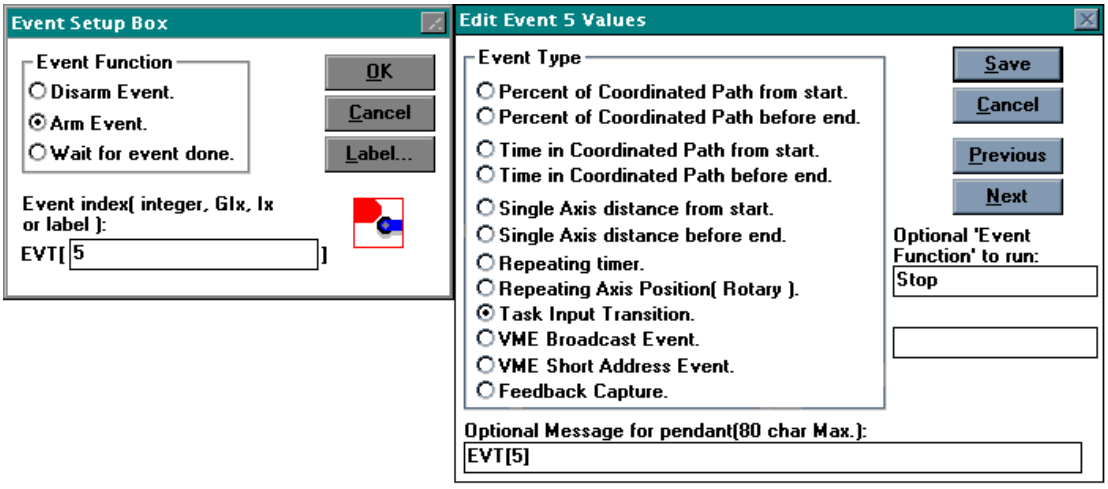


Interrupt Input Events

Bit 9 in each Task's Control Register is reserved as an Event Interrupt Input for the task. Each low-to-high transition of this input can trigger an event in the corresponding task. An Interrupt Input type event permits triggering an event function from an external input. The event number to be executed is specified in the “Event” icon on the “Coord” icon palette. After the user program is compiled and downloaded to the CLC, the event type, task input transition, and event function must be specified in the event table. The event table can be loaded by selecting “Events” under the “Data” menu in VisualMotion.

The CLC scans the input every 2ms and queues an event upon a low-to-high transition. The event function will take priority over the user tasks, allowing quick response to an external input.

The I/O mapper can be used to invert the logic of the interrupt input, or to direct other external inputs to the Task Control Register's Event Interrupt bit. Logic in the event function can then scan the multiple inputs to determine the source of the interrupt.



Extended External I/O Events

Extended User Inputs (EUIs) extend Task A's external input interrupt events from one to seventeen. The additional 16 are brought in through:

Register 88, the user external input register (USER_XI_REG)

Register 89, the user external output register (USER_XO_REG)

Tasks B, C, and D have only one external input interrupt event each. ***Refer to the Event Table description and Register 88-89 for more information.***

VME Events

The VME version of the CLC supports the ability to trigger an event on one or more cards on the VME bus.

Short Address Event

Using the VME short address space, a single event can be triggered on one other card.

A master device triggers the event by reading a byte from the location of mailbox 4 (A16:D8). The read generates a local interrupt on the CLC card, scheduling the proper task event handler to run. If the target CLC card acknowledges the trigger, the uppermost bit of the read operation will be 0. If a previous interrupt generated by reading the mailbox is still pending, the uppermost bit of the data will be 1. The CLC's VME short address space mailbox 4 is located at an offset of 16 (0x00010) from the CLC card's base address.

A local VME event can be initiated by using an Icon Language VMEEvt icon, or by a Text Language VME/EVENT instruction.

VME Broadcast Event

Later versions of CLC hardware, which include the connector P2 for accessing VME extended address space, can trigger an event simultaneously across multiple cards.

The broadcast message specifies the task to which the event is bound, and the event number to run. The broadcast message generates a local interrupt on the receiving card(s) which schedules the proper task event handler to run. If any CLC cards addressed by the broadcast message cannot accept it because a card has a previous message pending, all of the addressed cards will refuse the message and generate an error.

The unit's broadcast message address is a bit mask specifying the units to be affected. If the bit corresponding to a given unit number is high, then the unit will respond to the message.

The CLC message broadcast area always occupies VME address space FA000000-FAFFFFFF (A32) even when broadcast messages are not used. No other VME devices on the bus may use this address space. The base address for calculating a broadcast message address is FA800000. Address lines A2-A17 are used to designate units 0 - 15.

The appropriate CLC card (unit) addresses are listed in the following table.

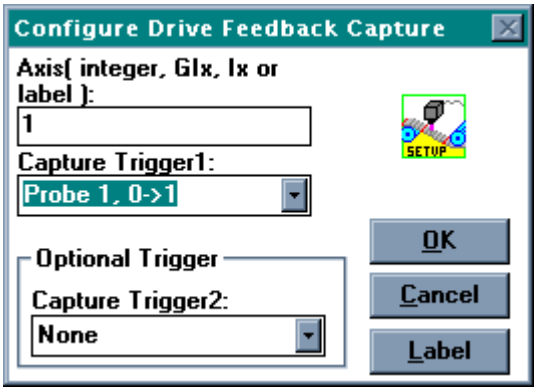
VME Unit	32-bit Address	VME Unit	32-bit Address
0	0xFA80 0004	8	0xFA80 0400
1	0xFA80 0008	9	0xFA80 0800
2	0xFA80 0010	10	0xFA80 1000
3	0xFA80 0020	11	0xFA80 2000
4	0xFA80 0040	12	0xFA80 4000
5	0xFA80 0080	13	0xFA80 8000
6	0xFA80 0100	14	0xFA81 0000
7	0xFA80 0200	15	0xFA82 0000

To select multiple units, OR the addresses. For example, to send a broadcast message to units 0 and 5, the master would write to VME address FA800084.

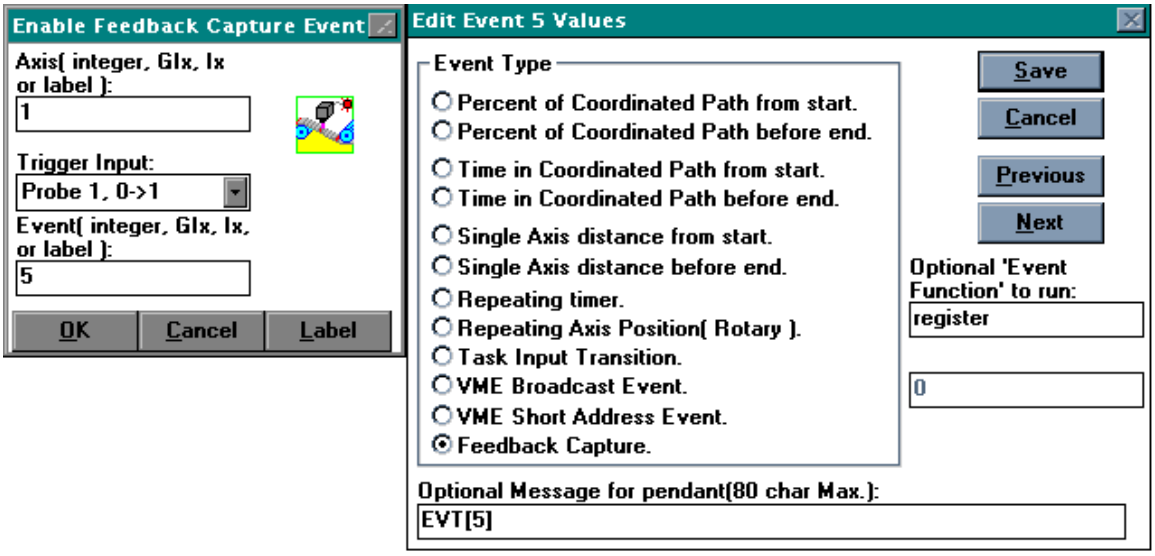
The data associated with a broadcast message contains the task ID and event index. Any VME data width formats can be used to write to the broadcast address, however, only the lower byte is used. The upper 2 bits of the broadcast message designate which task the event should be bound to. Tasks are assigned the following bit mask values: A = 00, B = 01, C = 10, and D = 11. The lower 6 bits of the message specify the event number (table index).

Feedback Capture (Probe)

The CLC uses the SERCOS Probe Functions and Real Time Bits along with the Event System to allow user programs to perform registration functions. The Indramat DDS drive provides two probe inputs that can be used for capturing the feedback position. Diax02 drives capture the position with 250 µsec and Diax03 drives capture the position within 5 µsec. Upon either a positive or negative transition of a probe input, the DDS captures the position into the cyclic data. Since the captured feedback positions must be included in the SERCOS cyclic data telegram, the probe setup icon must be included in the user program for each drive that will use the probe function.



The probe event must be armed each time it is to be executed. Typically, probes are used to detect registration marks on material. By controlling when the probe is armed, other printing on the material can be filtered out. When the position is captured, the drive signals the CLC with a real time bit in the SERCOS telegram. When the CLC detects a change in the real time bit, it can execute an optional event function. The event number to execute is specified in the “ProbeEvt" icon. After the user program is compiled and downloaded to the CLC, the event type, feedback capture, and event function to execute must be specified in the event table. The event table can be loaded by selecting “Events” under the “Data” menu in VisualMotion.



Event Tables

See Event Table and Appendix B. Direct ASCII Communications

5.4 Subroutines

Subroutines can be used to pass function arguments and can include local variables. This eliminates the need the keep and maintain temporary variables and allows user programs to be structured more effectively. Subroutines can also be re-used by more than one task.

The Sequencer uses subroutines to perform data driven programming that allows function arguments to be adjusted without changing the program.

Function Arguments and Local Variables

Function arguments are stack based variables passed to a subroutine. The setup dialog box for the Sub icon has entries for the function arguments. Up to 5 arguments may be passed to a subroutine. The arguments can be constants or variables (float, integer, ABS index or REL index). The function arguments need to be declared in the subroutine Start icon dialog box. A minimum and maximum value must also be entered for each of these arguments. This limits the range of argument values used during Sequencer editing from the teach pendant or the Sequencer interface.

Function arguments are not allowed in tasks or events, however, an optional return argument may be passed back to a task from a subroutine. A **Return Argument** is an optional single argument that may be returned from a subroutine. The setup dialog of the ***Finish icon*** of the subroutine screen has an entry for the return argument, it may be a constant or variable. The setup dialog for the ***Sub Icon*** has an entry for a variable to receive the return argument, if no variable is entered the returned argument is discarded. A return argument is a convenient way to “get position”, utilizing a common subroutine from more than one task

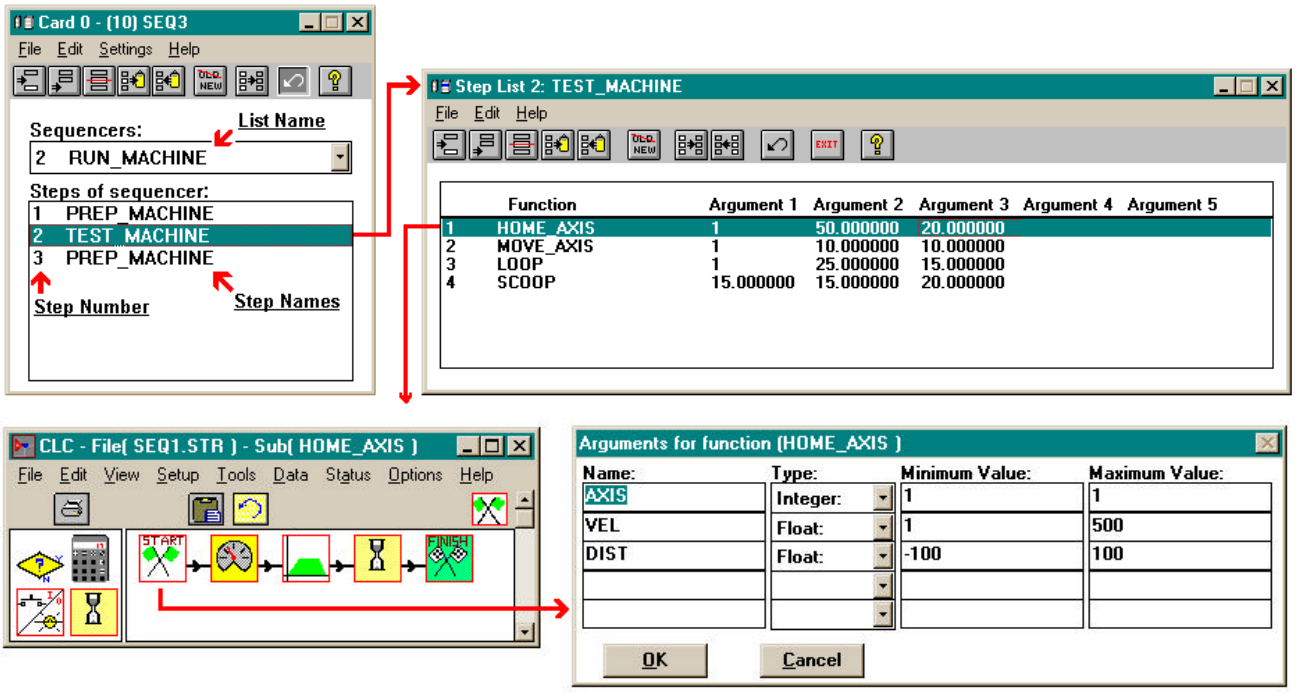
Local variables are stack based variables used in a function. The total number of local variables and function arguments is limited to 16. Local variables, floats and integers are initialized to zero on each entry into the function. This type of variable can be used to store temporary results in a re-entrant function called on by more than one task. Local variables can also be used to avoid possible conflicts with program or global variables between tasks.

5.5 Sequencer

The CLC Sequencer provides the user with a facility for making machine operational changes without having to edit, recompile and download a program to the CLC Card. Programs can be built or modified using predefined and tested functions *or subroutines* that are tailored to the machine. Each function can contain up to five argument values. Once these functions are in placed within a program a Sequencer can execute them in any order. To make a change, the user can simply add functions to a Sequencer Step List, then edit the function arguments as required. When the program runs, the CLC executes each function sequentially, in the order it is listed.

A Sequencer List is essentially a "to do" list. This list is broken down into individual steps. Each step contains a list of one or more functions with up to five arguments per function. The order of program flow sequentially follows this "to do" list through each of its steps and functions. The function *or subroutine* will then run with the values of the function arguments given in the Step List. Each Sequencer List can contain up to 30 steps.

SEQUENCER LIST

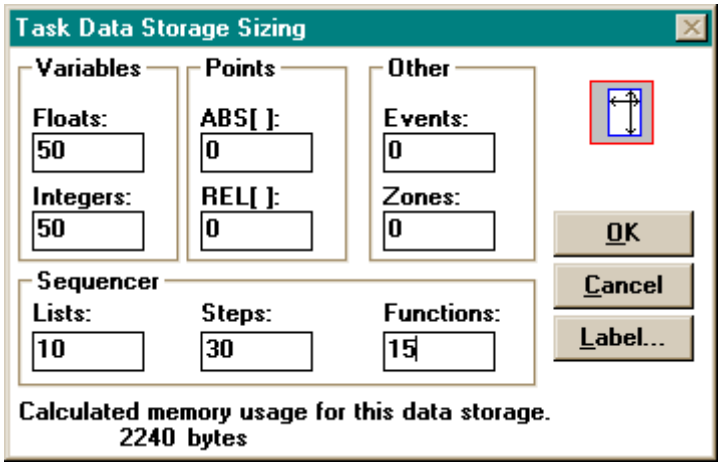


FUNCTION - Sub(HOME_AXIS)

START ICON - Dialog Box

The "Callable from Sequence" access flag must be set in the ***START icon*** of the subroutine before that function can be added to a Step List. Function arguments must also be defined before they can be used. ***See SUB icon***.

The maximum number of *different* Sequencer lists and steps is configurable in the ***SIZE* icon**. If the same list or step is used more than once, it will only count as one towards the total. The number of functions refers to the total number of all functions executed and is independant of the number of Lists and Steps. Every function in a step counts, even if it is the same function used more than once. Sequencer lists, steps and functions are identified by a unique user determined name which can be up to twenty characters long.



Any Sequencer List or Step can be edited with the teach pendant and/or with VM. When using VM, the Sequencer editor under the Data menu can be used to edit or add to a Sequencer list. If the teach pendant is used the resulting software will be upwardly compatible with existing programs.

The Sequencer list can also be initially setup in the program within the ***Sequencer icon***. If a Sequencer is initialized and setup in a program, any changes made to it with the Sequencer editor will be overwritten when the icon is executed.

Single Stepping a Sequencer Step

Bit 12 in the Task_Control_Registers(2-5) is used to single step a Sequencer step. When set to 1, it instructs the CLC to stop program execution after each Sequencer step. Toggling the Cycle_Start (bit 6) will continue execution through the next step.

Single Stepping a Step Function

Bit 13 in the Task_Control_Registers(2-5) is used to single step a function in a step. When set to 1, it instructs the CLC to stop program execution after each step function. Toggling the Cycle_Start (bit 6) will continue execution through the next function.

5.6 Data and Expressions

Integers and Floats

Integers are signed or unsigned whole numbers, such as 5 or -3. Integers cannot have a fractional component. CLC integers are stored as a 32-bit "two's complement" value using the high bit for the sign. Integers, therefore, have a range of -2147483648 decimal (80000000 hex) to +2147483647 (7FFFFFFF).

Floating point values possess three components. A signed integer component (the "characteristic"), an unsigned decimal ("mantissa") component, and an exponential component (the "exponent") using the base 10. Even if you do not specify a decimal or exponential component, floats are stored in full floating point format. Floating point numbers are typically specified by using a decimal point. (i.e., "5" is an integer, "5.0" is floating point.)

The CLC stores integer and floating point variables as 32 bit double words to preserve word addressing on even memory locations. Thirty-two bit addressing also simplifies operations on the CLC's I/O registers. All variables are 32-bit values; CLC I/O registers are 16 bit values.

Constants and Variables

CLC programs permit both constants and variables in user programs. These are useful in that they allow storage of information that is application specific, such as point locations, but changing them does not require modifications to associated motion programs.

The tasks and subroutines within a CLC program share a common storage area. Therefore, all tasks and subroutines access the same data. In addition, the CLC provides integer and floating point global system variables that may be used to share values with other system devices that can access global memory. (i.e., multiple CLC cards on a VME bus.)

A constant may be any valid integer or floating point literal constant (e.g., a user specified number, such as 3.14159). Constants cannot be modified by the program during program execution.

Integer variables are identified by an upper case ASCII character I immediately followed by a string of decimal digits or a symbolic name enclosed in square brackets, such as I[01] or I[15]. Integers are stored in an integer variable table in the CLC memory. Floating Point variables are identified by an upper case ASCII character F immediately followed by a string of decimal digits or a symbolic name enclosed within square brackets, as in F[01] or F[23]. Floating point variables are stored in a floating point variable table in the CLC memory.

The CLC compiler also accepts a special case for constant indices to the integer or floating point tables. In this case the square brackets are not required. The table entry is addressed by adding an integer constant immediately after the I or F variable identifier. Each integer or floating point constant or variable requires 4 bytes (32 bits) of CLC memory.

Examples:

Variable	**Variable Type**
I[15]	;integer variable, formal form
I37	;integer variable, short form (integer constant)
I[Ipointer1]	;symbolic integer variable (Ipointer1 must evaluate to an integer)
GI[15]	;global integer variable, formal form
GI37	;global integer variable, short form (integer constant)
GI[Ipointer1]	;global symbolic integer variable (Ipointer1 must evaluate to an integer)
F[01]	;floating point variable
F99	;floating point variable, short form
F[Fpointer3]	;symbolic floating point variable (Fpointer3 must evaluate to an integer)
GF[01]	;global floating point variable
GF99	;global floating point variable, short form
GF[Fpointer3]	;global symbolic float (Fpointer3 must evaluate to an integer)

Global Variables

The global variables consist of a contiguous array of 256 floating point and 256 integer variables. The global variables are accessed as arrays of 4 byte values beginning at specified base addresses. The specific addresses depend on the type of CLC card. Global variables reside in CLC RAM and are not retained during power-off. Global variables in user programs are not automatically initialized by the CLC compiler. It is the programmer's responsibility to explicitly initialize global variables in a program. If values need to be saved, they may be copied to standard variables which are battery-backed, provided sufficient space has been allocated.

Because the variable tables are common areas, they can be read or written at any time by any of the user tasks, subroutines or event functions. Therefore, the common data areas are used to pass values between tasks or subroutines. Global variables may also be used to exchange values between external components of a CLC system that are capable of accessing the global memory area. However, using variables to communicate program values directly can have unexpected results. Because the CLC's multitasking can be event driven, systems cannot guarantee the order of event function execution. Use caution when directly exchanging program data by writing to the same variable from more than one task or subroutine.

Depending on the occurrence and number of pending events, you cannot always be sure of the exact time that variables will be read or written. User programs may assign bits within one or more variables as "wait" flags or "semaphores" controlling program branching, thereby synchronizing critical portions of the otherwise independent tasks and programs.

Expressions

An expression is used to calculate or modify variables. Standard algebraic syntax is used to write a mathematical expression within a CLC program. An expression may use any combination of constants, variables, operators, functions, and parenthesis. The value to the left of the "=" is set to the calculated value of all elements of the expression on the right. The right side of the expression may include up to 16 constants, variables, parentheses, operators, and functions.

Evaluation and Precedence

The right side of an expression is evaluated sequentially from left to right. There is no operator precedence. If necessary, balanced parentheses should be used to insure that your expression is evaluated in the order that you intend. For example: RATIO = 4 + 3 * 5 - 1

evaluates RATIO as 4+3=7, then 7*5=35, then 35-1=34, while RATIO = (4 + 3) * (5 - 1) evaluates RATIO to 28.

Type Conversion

If an expression uses both integer and floating data types, the CLC compiler converts all operands to the floating data type before evaluating the expression. After evaluation, the result is converted to the data type of the destination (left side of the expression).

Mathematical and Logical Operators

The operators available for use in expressions are listed in the following table.

Arithmetic operators	
=	assignment (assigns the evaluated expression on the right side of the "=" operator to the variable on left side of the operator)
+	addition (floating point or integer)
-	subtraction (floating point or integer)
*	multiplication (floating point or integer)
/	division (floating point or integer)
Logical operators **Note: Logical operations are limited to unsigned integers.**	
&	logical bitwise AND of two integers
|	logical bitwise Inclusive OR or two integers (101 | 011 is 111)
^	logical bitwise "exclusive or" of two integers (101 ^ 110 is 011)
not	logical bitwise inversion of an int (changes "0s" to "1s" and "1s" to "0s")
<<n	Int shift "n" bits left (1-16 bits), low bits padded with 0, high bits are lost
>>n	Int shift "n" bits right (1-16 bits), high bits padded with 0, low bits are lost
Transcendental functions **Note:** ***All transcendental functions are in radians.***	
sin(n)	returns the floating point sine of an integer or float
cos(n)	returns the floating point cosine of an integer or float
tan(n)	returns the floating point tangent of an integer or float
arcsin(n)	returns the floating point arcsine of an integer or float
arcos(n)	returns the floating point arccosine of an integer or float
arctan(n)	returns the floating point arctangent of an integer or float
ln(n)	returns the floating point natural logarithm (base e) of an integer or float
log(n)	returns the floating point logarithm (base 10) of an integer or float
Exponential functions**:**	
n**p	returns the floating point value of "n" raised to the "p" power
sqrt(n)	returns the floating point square root of an integer or float
Conversion functions**:**	
%	modulus (the remainder or fractional portion) of the result of the division of 2 integers or floating point numbers
int(n)	returns the integer portion of a floating point value as an integer
float(n)	returns a floating point value equal to an integer
frac(n)	returns a floating point value equal to an integer
absolute(n)	converts a positive or negative integer or float to a positive integer
bintoBCD(n)	converts a binary value to a packed BCD integer
BCDtobin(n)	converts a packed BCD to a binary value

Expression Examples:

```
I03 = I03 + 2
```

Adds 2 to the current value of the integer variable I03 and replaces the current value of I03.

```
F01 = 3.14
```

Replaces the current value of floating point variable F01 with the constant 3.14.

```
F33 = COS(F33)
```

Replaces F33 with the cosine of F33

```
F15 = (F10 + F11)*(F12 - F13)
```

Replaces the current value of F15 with the result of the multiplication of the value of F10 plus F11, times the value of F13 subtracted from F12.

5.7 Tables

The CLC stores program variable information and data in separate areas of memory called tables, also referred to as arrays.

CLC integer and floating point variables are maintained as one dimensional arrays. You can think of a one dimensional array as a simple list, or as a spreadsheet that contains series of rows with only one column. These arrays are referred to as the integer and floating point tables. Each entry requires 4 bytes of CLC memory.

As previously described, a variable table entry is addressed by using an "I" or "F" prefix before an integer value that is within the range of the respective table. For example, I[15] accesses the 15th entry of the integer table, or F[43] accesses floating point table entry 43. This is "direct" addressing of a variable. The integer may be thought of as an "index" into the table, with the base address of the table being the first entry, i.e. the integer "1".

Indirect Addressing

A calculation instruction may also access a variable or table entry indirectly by enclosing an integer variable within square brackets. This format allows you to use programmed calculations to dynamically control the array index. As a result, the program can use an I/O input value or calculation result to determine which variable or table entry is used. For example, F[I[3]] (or F[I3]) returns the value of the floating point variable table entry addressed (or "indexed") by the contents of the third integer table entry, I[3]. By writing a program that changes the contents of I[3] you can obtain a value from a table of constants for further calculation.

In addition to the single dimensioned arrays used for the variable tables, the CLC also uses several two dimensional arrays. The absolute and relative point tables are two dimensional arrays that contain information about a coordinated motion path. The event table which contains information and parameters for the event functions is also two dimensional.

A row of the two dimensional tables, absolute and relative point, and event, may be accessed using an index enclosed within square brackets "[ ]" in the same manner as the single dimension variables. Accessing a two dimensional table element uses the dot operator (".") to select a column element within the selected row.

An element of the absolute or relative point tables, or the event table is accessed using an ABS, REL, or EVT prefix to a row index enclosed within square brackets, immediately followed by the array dot operator (".") and a row element identifier.

For example: **`ABS[n].x`** **`REL[n].x`** **`EVT[n].r,`** where ABS or REL or EVT accesses the absolute point, or relative point, or event table, "n" is a valid integer constant or variable or equivalent label, and "x" is a table element identifier. Blank spaces are not allowed.

The integer value contained within the brackets may be thought of as an index to one of the rows of the table, while the letter following the dot operator indexes to the column position within that row. You may use a symbolic name or integer variable within the square brackets to access a table row, provided that the name evaluates to an integer value equal to a valid table entry.

For example: **`ABS[I[n]].x`** **`EVT[event3].x`**

Event Table

Events in a CLC program are stored in the CLC Event Table:

Event #	s	t	d	a	f	m
01	Active	Time	Start	10	In_Zone	Message 00
02	Inactive	Distance	End	10	Gripper_On	Message 01
03	Active	Distance	Start	50	Gripper_On	Message 02
04	Inactive	Distance	End	25	Gripper_Off	Message 03
05	Undefined					
-	-	-	-	-	-	-
nnn	Undefined					

The elements of the Event Table are defined in the following table:

s	**status**	0 = inactive 1 = queued 2 = pending 3 = executing 4 = done
t	**type**	0 = event inactive 1 = repeating timer 2 = time on coordinated motion path 3 = distance on coordinated motion path 4 = single axis distance 5 = repeating axis position 6 = task external interrupt input 7 = VME broadcast interrupt 8 = VME short address (mailbox) interrupt 9 = axis feedback capture 10 = Task A extended external interrupt
d	**reference direction**	0 = start of segment 1 = end of segment
a	**argument (trigger)**	a numeric value (in milliseconds if timed event) (a % of the segment distance if coordinated motion) (degrees if repeating axis position event) (contains the probe position read from the drive if feedback capture event is selected) (bit number if extended external interrupt)
f	**event function mark**	Name of subroutine to be called when event is triggered
m	**message**	ASCII text string for a descriptive message

CLC Path Planner

The CLC's path planner interprets coordinated motion commands for all user program tasks and associated events. Using the program specified points in the absolute and relative point tables, the path planner continually calculates the position information that must be sent to each drive to maintain coordination between the multiple axes. The path planner is one of the highest priority CLC executive tasks.

The path planner continuously monitors the difference between the current system position and the planned or target position that will result from the user program. The distance that the path planner will look ahead of the current position is set by the CLC's look ahead distance parameter T-0-0023.

Geometry Segments and Paths

The CLC defines multi-axis coordinated motion in terms of a path composed of standard straight line and circular geometry segments. Two points define a line, three points define a circle. The path combines these standard geometry segments so that the start of the next segment begins at the end of the previous segment.

The CLC always calculates a motion path using a series of points taken from the Absolute or Relative Point tables. The absolute or relative table entry that is used for the end point of a segment contains the rate profile parameters (speed, acceleration, deceleration and jerk) used by the path planner to construct the rate profile for that segment. Starting or middle points are not used to construct the profile, only the table point entry rate profile parameters for the end or target point of the segment are used.

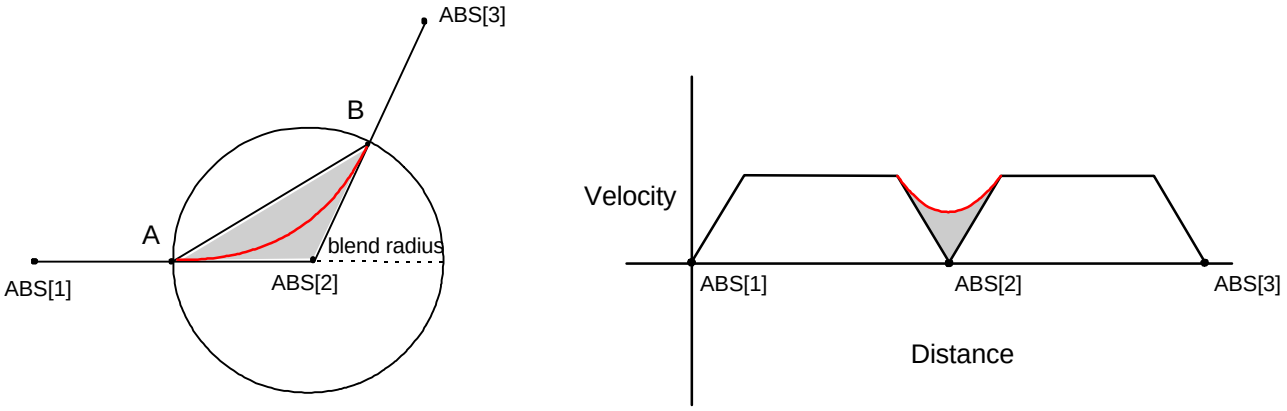
Blend Segments

Blend segments allow motion to continue across one standard geometry segment to another without stopping. They are used to reduce the path planner cycle time and also provide a means for optimal path shaping. The path planner uses a blend radius value to automatically compute the blend segment (*See Section Absolute Point Table*).

The following illustration shows two standard geometry segments between ABS[1] and ABS[2], and between ABS[2] and ABS [3]. The blend radius determines three points within the segments. The first one is the connection point between the two standard segments (ABS[2]). The second and third points (A and B) are the points on each segment that are the blend radius distance away from the first point (ABS[2]).

These three points define a circle in (x, y, z) space. The intersection of the circle and the standard segments form a wedge between the first and second segments. The blend is computed inside the wedge to start at point A and end at point B. The exact shape of the blend is determined by the speed, radius, acceleration, deceleration and jerk of the standard segments before and after it.

The profile on the right shows how the blend radius affects velocity within the geometric segments. Without the blend segment the velocity would drop out at ABS[2]. When a blend segment is added the velocity at ABS[2] will increase as the blend radius increases.



Speed

The point table entry for speed defines the velocity of motion through the segment as a percentage of the maximum velocity (parameter T-0-0020) for the axis. The speed within a blend segment is still determined by the length of the blend radius.

Acceleration/Deceleration

CLC acceleration and deceleration for a single non-coordinated axis is set as an acceleration constant, in units per second per second, using the AXIS/INITIALIZE command. Acceleration is set as a percentage of the maximum acceleration (parameter T-0-0021); deceleration is set as a percentage of the maximum deceleration (parameter T-0-0022). The deceleration for the axis is the inverse of the specified acceleration. Single axis accel/decel rates are developed within the intelligent DDS drive itself and require minimal processing by the CLC.

Acceleration and deceleration for multi-axis coordinated motion are set in corresponding entries in the absolute or relative point tables as a percentage of the task's maximum rates. The maximum rates are established through user-set parameter values.

Jerk Limiting

The CLC provides two forms of jerk limiting. The simplest, used with a single axis, is 'S' curve limiting and is generated within the intelligent DDS drive itself. More sophisticated jerk limiting is provided by the CLC path planner for multiple coordinated axes.

Jerk limiting should only be used for start and stop movements.

Important Note: The jerk limit should be set to zero when blending one segment into another.

The jerk limit affects rate limiting. For example, with jerk limiting set to 100% (total s-shaping), acceleration and deceleration limiting occur at 50% of the point table entries. For optimum performance, the jerk limit should be set to a minimum so that there is no acceleration or deceleration limiting.

Absolute Point Table

Absolute points, used to define geometric path segments for coordinated motion, are stored in an absolute point table separate from the executable program code. The absolute point table is also saved in a point file on the Host PC. Points may be defined in motion programs, or directly entered in the point table if the program resides in the CLC.

Example Absolute Point Table:

Point	X	Y	Z	Blend	Speed	Accel	Decel	Jerk	E1	E2	E3	E4	r	p	w	e
001	99.0	-333.2	95.5	10	20	100	10	90	5	0	0	0	30	0	0	0
002	2.0	33.2	-60.5	5	67	50	10	80	1	7	0	0	60	0	0	0
003	2.0	33.2	-60.5	1	8	60	95	80	0	0	0	0	90	0	0	0
|	|	|	|	|	|	|	|	|	|	|	|	|	60	0	0	0
nnn-2	0.0	5.9	95.5	5	10	20	30	70	5	6	8	9	30	0	0	0
nnn-1	00.0	-333.2	95.5	10	40	99	40	50	4	0	0	0	15	0	0	0
nnn	00.0	-333.2	95.5	10	9	99	30	55	3	0	0	0	10	0	0	0

The elements of the Absolute Point Table are defined in the following table:

Subclass	Description	Type	Size
x	x cartesian coordinate	Floating	32-bit
y	y cartesian coordinate	Floating	32-bit
z	z cartesian coordinate	Floating	32-bit
b	blend radius	Floating	32-bit
s	% of maximum speed	Integer	8-bit
a	% of maximum acceleration	Integer	8-bit
d	% of maximum deceleration	Integer	8-bit
j	jerk limiting %	Integer	8-bit
E1-E4	event ID number	Integer	16-bit
r	degrees of roll or rate (if kinematic #=8)	Floating	32-bit
p	degrees of pitch	Floating	32-bit
w	degrees of yaw	Floating	32-bit
e	elbow state/coordinated axis mask	Integer	8-bit

Note: *The "r" subclass will refer to "rate" and not "degrees of roll" if the kinematic # = 8. See the Axis Icon description.*

Relative Point Table

Relative points, used to define geometric path segments for coordinated motion, are stored in a relative point table separate from the CLC user program. The relative point table is also saved in its own file on the personal computer (PC) Host development system.

During the processing of relative moves, events and the velocity profile parameters associated with the relative point being processed are active. The events and the velocity profile parameters associated with the absolute or reference point or origin point are not used. The events from the relative table are triggered; events from the absolute point table are ignored.

Example Relative Point Table:

Point	X	Y	Z	Blend	Speed	Accel	Decel	Jerk	E1	E2	E3	E4	r	p	w	e
001	99.0	-333.2	95.5	5	20	100	1	90	5	0	0	0	30	0	0	0
002	2.0	33.2	-60.5	20	70	50	10	80	1	7	0	0	60	0	0	0
003	2.0	33.2	-60.5	50	80	60	20	80	0	0	0	0	90	0	0	0
|	|	|	|	|	|	|	|	|	|	|	|	|	60	0	0	0
nnn-2	0.0	5.9	95.5	50	10	20	30	70	5	6	8	9	30	0	0	0
nnn-1	00.0	-333.2	95.5	30	40	10	40	50	4	0	0	0	15	0	0	0
nnn	00.0	-333.2	95.5	5	55	10	50	55	3	0	0	0	10	0	0	0

The elements of the relative point table are defined in the same way as those of the absolute point table.

Zone Protection Table

The CLC's zone tables provide a method of detecting and preventing motion through one or more three-dimensional rectangular volumes. Protective zones are typically used to prevent motion that would otherwise be attempted through supporting columns of a robot, or would damage clamps or fixtures. Zones are also used to prevent motion through spaces near a machine that may be occupied by an operator or other personnel. Zones are active during jogging as well as coordinated motion.

A prohibited-zone volume is described by specifying pairs of three dimensional coordinates that locate diagonally opposite corners of the volume. Each table entry consists of a status bit and the two sets of x, y and z coordinates. The status bit permits independently activating/deactivating individual zones within the table. Each of the four tasks (A, B, C and D) within a user program using zones maintains a separate zone table for the task. Memory for the task's zone table must be allocated by a Text Language DATA/SIZE instruction, or Visual Motion's SIZE icon, within the task.

Zone Detection

Detection of an instruction that may violate a zone is accomplished in the path planner, permitting zone violations to be detected before motion starts. However, detection is related to the user specified look-ahead distance.

For circular motion, the radius is used to determine zone interference. The total length of the circle is computed and interference calculations are taken at exact intervals along the circumference of the circle.

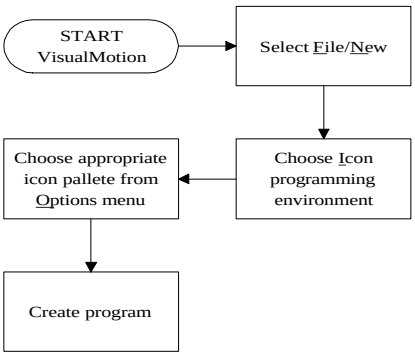
Joint moves are tested using a spherical volume instead of the rectangular volume. The prohibited volume is a sphere enclosing the rectangular area specified by the zone table coordinate pairs. In other words, the zone volume is a sphere with a radius equal to the distance from the center of the rectangular volume to a corner of the rectangle.

Zone interference detected by the path planner stop motion while maintaining the system position on the path.

6 Icon Programming

6.1 Introduction

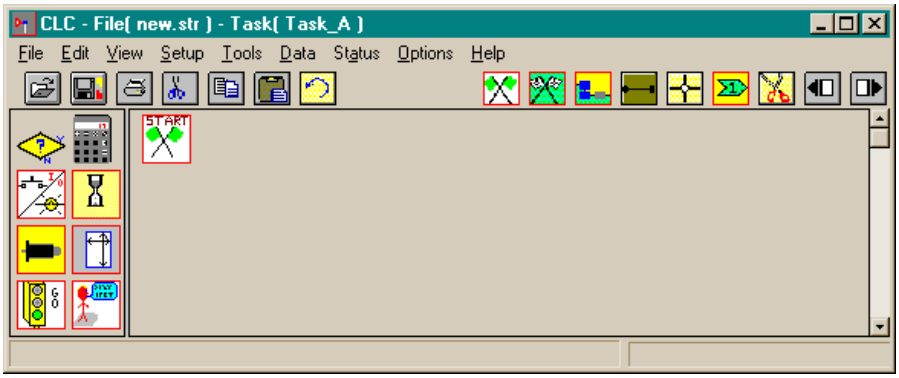
This chapter describes how to use icons and other VisualMotion commands to create motion programs, a process briefly illustrated in the flow diagram below.



The first section describes how icons are placed and connected in motion control programs. The second section contains descriptions of each icon, presented in alphabetical order

Working with VisualMotion's Icon Palettes and Buttons

The VisualMotion window displays a palette of programming icons. Four standard palettes are provided for single, coordinated, ELS and Utility icons. An icon in the palette is selected for placement by clicking it once with the left mouse button. The selected icon is placed on the VisualMotion workspace by positioning the cursor where you want the icon to appear and clicking once. A different icon may be selected for placement simply by clicking a new icon in the palette.



⇧**Icon Palette** ⇧**VisualMotion Workspace**

The VisualMotion window also displays a set of frequently used operational and icon buttons. These buttons will only appear on the screen when they are activated by a selection or an operation.

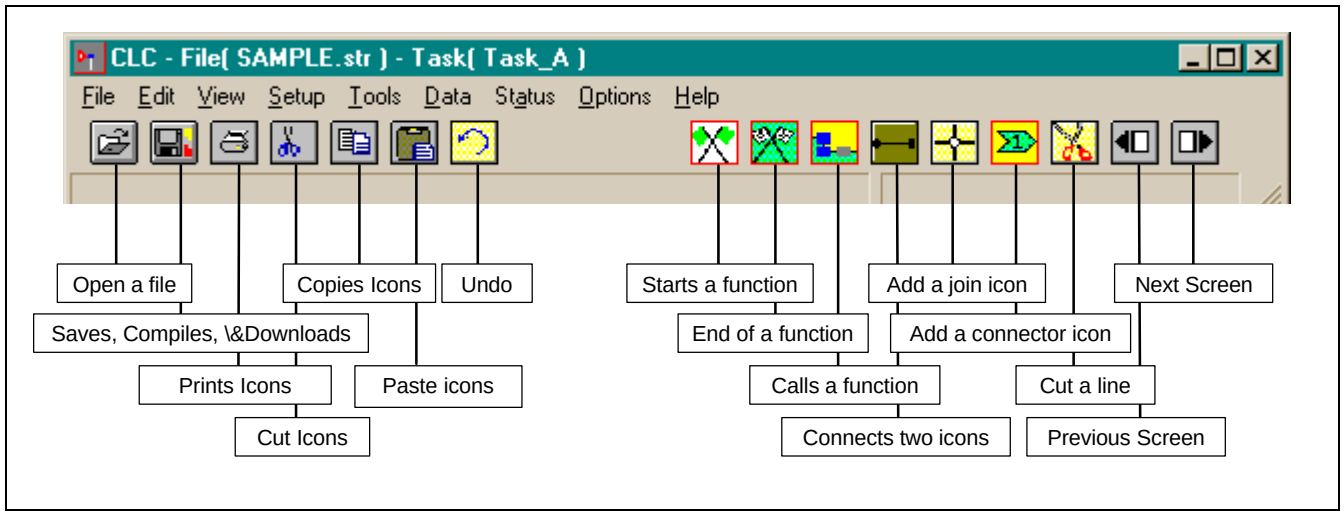
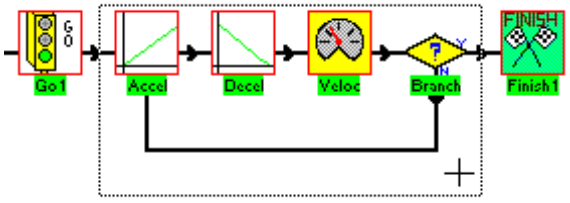
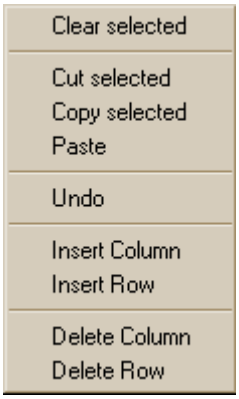


Figure 6-1: Visual Motion Program Menu and Icon Button Bar

VisualMotion requires a mouse with right and left push-buttons. (Although some mice have three buttons, VisualMotion ignores the center button). The left button is used for selecting single icons or lines; selecting commands from the pull-down menus; and positioning the cursor in dialog boxes for keyboard entry. In order to select multiple icons or lines, click and hold the left mouse button while drawing a selection box around a program section.



Once a selection has been made the cut and paste buttons can be used to delete or move one or more icons. In order to delete or clear an icon press the delete key or select delete for the edit menu. Once a selection has been cut or copied it is saved to the Windows clipboard. If the paste button is pressed the selection will appear on the screen and follow the cursor until a position for it is selected with the left mouse button.



The right mouse button is used for additional cut and paste operations. The **Clear**, **Cut** and **Copy** selected options operate the same as the described above. The **Undo** command will only undo the last operation performed.

The **Insert Column** and **Row** commands allow you to move several icons at once in order to add a new section to the program. In order to Insert a Column or a Row, an icon or a blank section in the workspace must be selected first. The **Insert Column** command will move all the icons above, below and to the right of the selection, over one space to the right. The **Insert Row** command will move all the icons to the left, right and below the selection, down one space. The **Delete Column** and **Row** commands will delete all the icons above and below, or to the left and right of the selection.

Connecting Icons

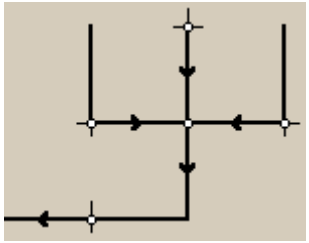
After you have placed a number of icons on the workspace, they must be connected to indicate the program flow. Most icons have a maximum of three possible inputs and one output. The exception is the Branch icon which has two outputs.

To draw a line, select the Line icon from the Icon Button Bar. Position the cursor on the first icon that you wish to connect and click. A rectangle appears, surrounding the icon. Move the cursor to the destination icon where the line is to end and click again. Visual Motion automatically draws a line from the first to the second icon, using square corners where appropriate. Arrows on the line indicate the direction of program execution. You may continue this process, connecting icon to icon, without re-selecting the Line icon.

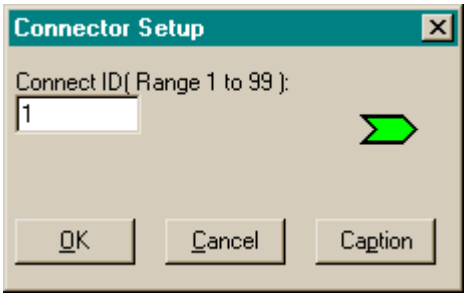
You may wish to manually route an interconnect to provide room for additional icons at a later time. Under some circumstances, the Line icon's auto-routing may fail to route an interconnecting line, displaying the "Try connecting adjacent lines." dialog box. Lines may be drawn manually by sequentially clicking on adjacent squares on the invisible workspace grid. A manually placed line may not cross another line, attempting to do so displays an error box.

A line connecting two icons may be deleted by using the ***Scissor*** icon from the *Icon Button Bar*. Simply select the Scissor icon, position the Scissor over the line to be deleted and press the left mouse button.

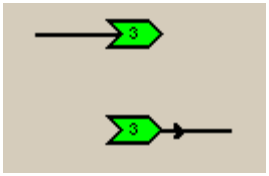
A ***Join*** icon makes it possible to connect one line to another from different directions.



The **Add a connector** icon allows the path of a program to flow between two points that are not connected by a line. This allows complex programs to logistically fit in a relatively small workspace.



When the icon is selected and placed in the workspace a Connector Setup dialog box will open and allow you to assign the icon with a Connect ID number (from 1 to 99).



Another **Add a connector** icon with the same Connect ID number can be placed anywhere within the program. The order of program execution will jump from the first connector icon to the second one.

The **Previous screen** and **Next screen** buttons provide a shortcut to jump between Program Tasks, Subroutines and function screens. ***See Programming Concepts and the View Menu under Visual Motion Menu Commands.***

Icon Labels and Comments

Each Icon has a label which can be displayed when "Icon Labels" is selected from the ***Options Menu***. Descriptive comments up to 80 characters long may also be assigned to each icon in a program. When "Icon Comments" is selected under the Options Menu, the comments will appears in a pop-up window as the cursor passes over the icon.

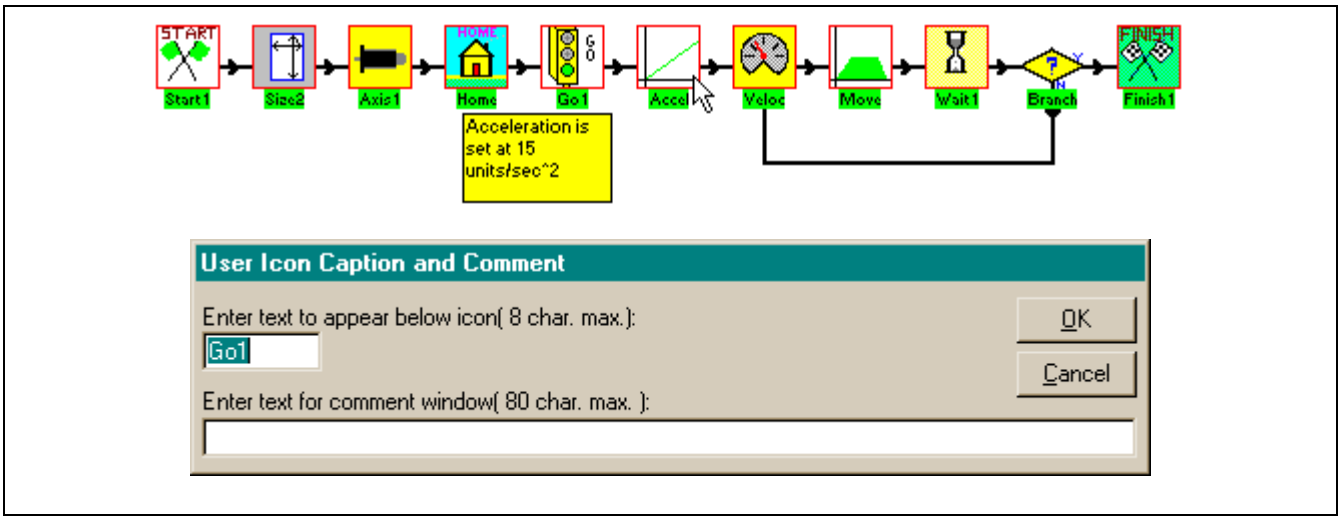
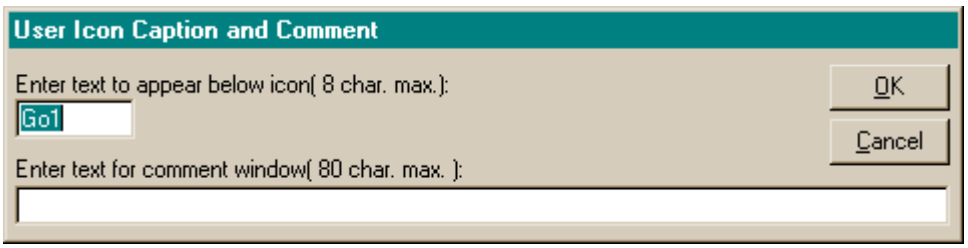


Figure 6-2: Icon Captions and Comments

The comments can be entered by pressing the "Caption…" button within the icon dialog box. They are intended to be used as a place to document the purpose of the icon relative to the program. The Caption under each icon can also be changed independently.


User Icon Caption and Comment
Enter text to appear below icon(8 char. max.):
Go1
OK
Cancel
Enter text for comment window(80 char. max.):

User Defined Labels

User Defined Labels pop-up dialog box is accessible from entry fields by setting the focus in the field and then double clicking the left mouse button outside the field. ***See Chapter 4. VM Menu Commands - Labels***.

6.2 CLC Icons

The VisualMotion window displays a palette of programming icons. Four standard palettes are provided for single, coordinated, ELS and Utility icons. An icon in the palette is selected for placement by clicking it once with the left mouse button. The selected icon is placed on the VisualMotion workspace by positioning the cursor where you want the icon to appear and clicking once. A different icon may be selected for placement simply by clicking a new icon in the palette. The current icon may be de-selected, freeing the workspace cursor, by re-clicking on the selected palette icon.

The CLC icons are illustrated below within their respective tool bars. Each icon is then listed in alphabetical order with functional descriptions that include example dialog boxes used for selecting options and data entry.

Single Axis Icon Palette

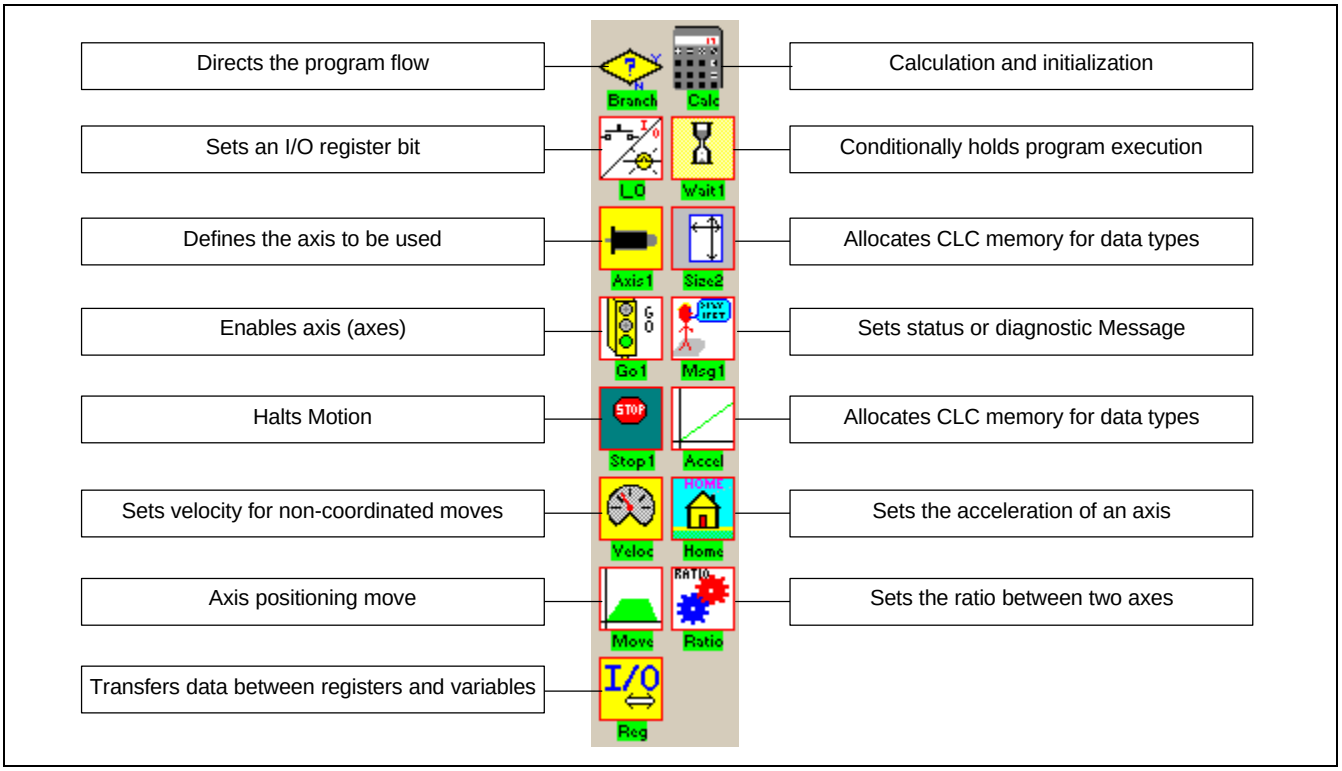


Figure 6-3: Icon Palette ‘Single’

Coordinated Motion Icon Palette

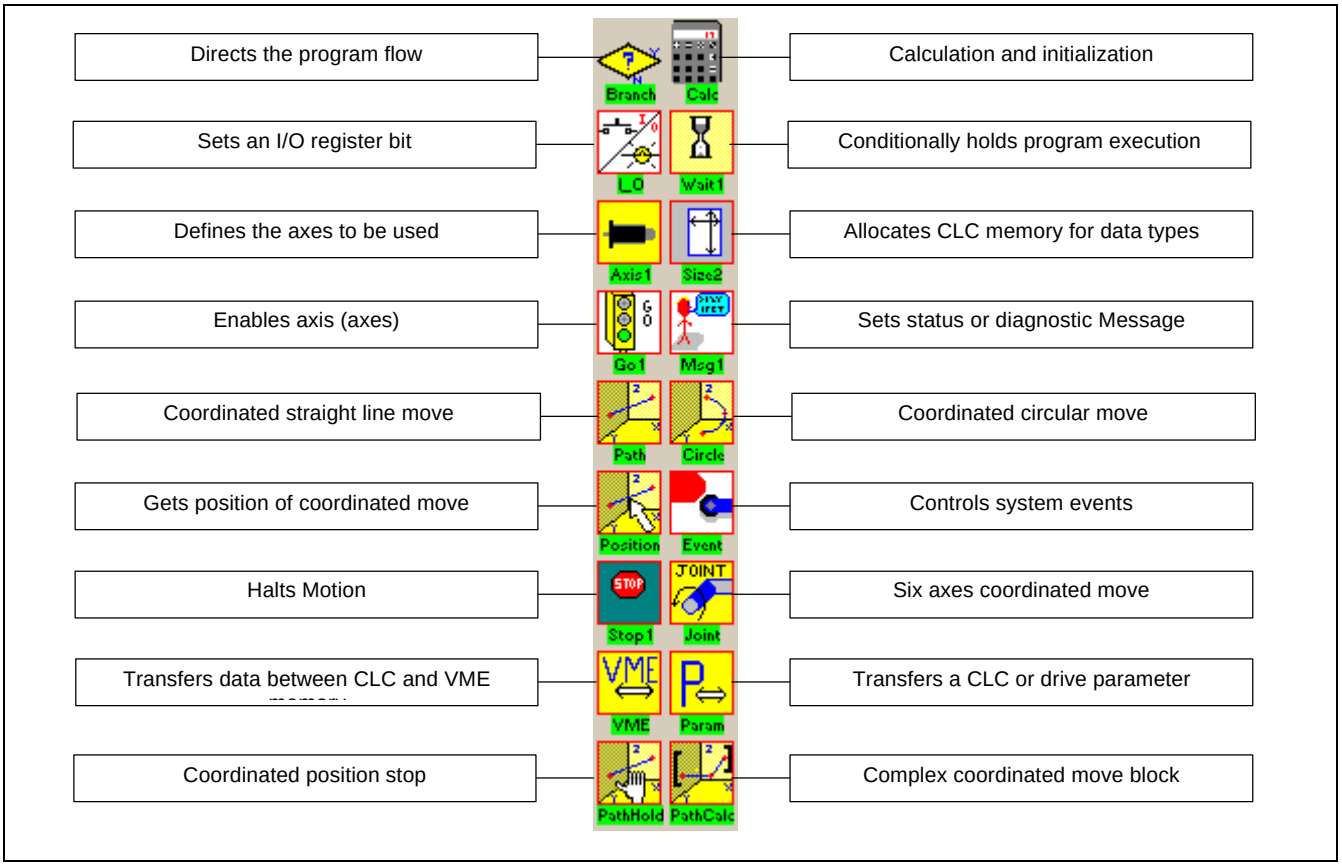


Figure 6-4: Icon Palette ‘Coord’

ELS Icon Palette

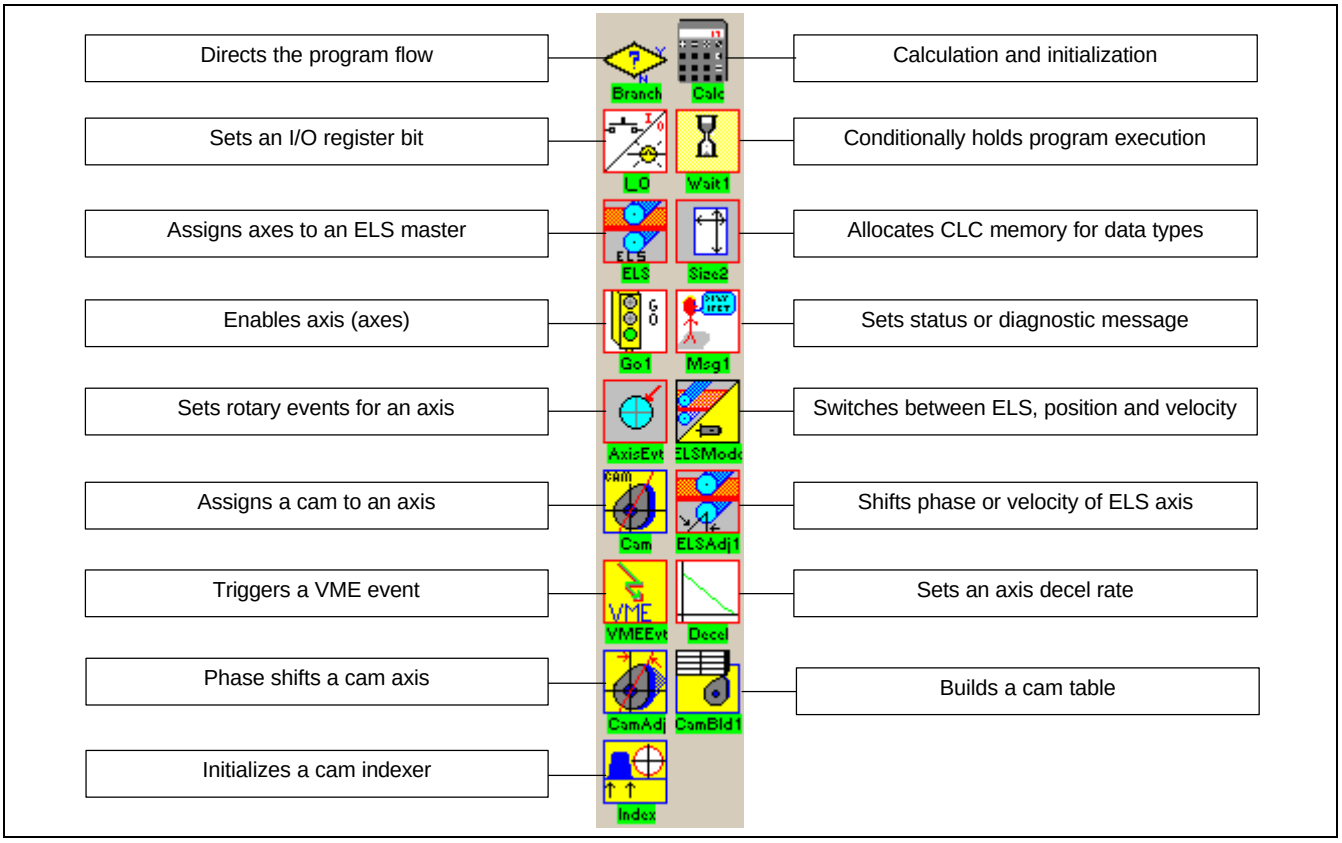


Figure 6-5: Icon Palette ‘ELS’

Utility Icon Palette

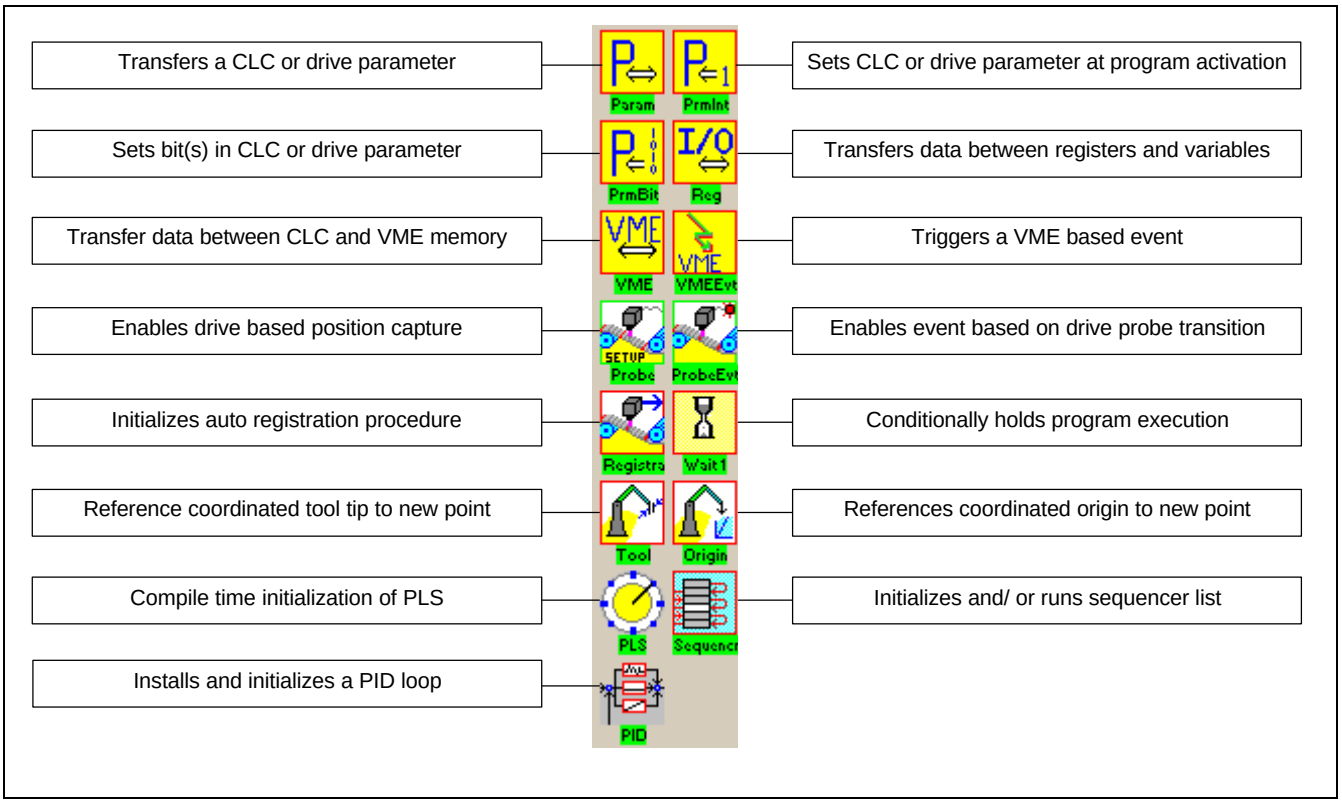


Figure 6-6: Icon Palette ‘Utility’

Accel

The Accel icon is used to set or change the acceleration of a single non-coordinated axis or the ELS Virtual Master (Axis 0).

Axis indicates the axis to accelerate. The axis may be entered as an integer constant, integer variable, global integer variable or an equivalent label.

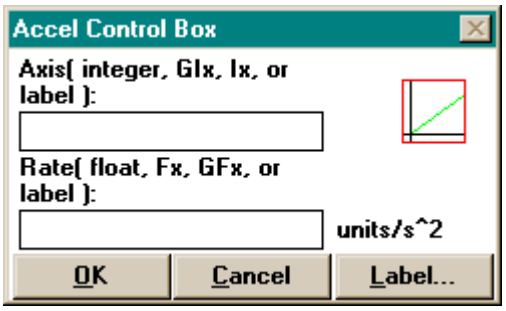
Rate specifies the acceleration rate in units per second per second. The entry may be a floating point constant, variable, global variable or an equivalent label. The variable must be greater than 0. A value <= 0 will generate an error “469 Axis D accel <= 0 or > maximum”.

Units:

Axis Definition

Single Axis Linear - inches	inches/s2
Single Axis Linear - mm	mm/s2
Single Axis Rotary	rads/s2
Velocity Mode	rads/s2
Virtual Master	rads/s2

The User Defined Labels pop-up dialog box is accessible from both data entry boxes.



Axis

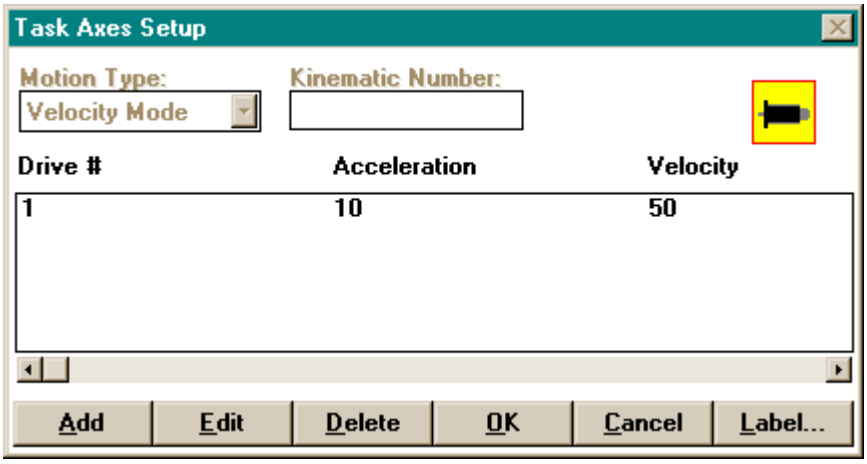
The Axis icon is used to specify drives that are assigned to the current task. The Axis icon can only be used to assign a drive within the four Tasks, it should not be used in a subroutine or event function.

Motion Type provides a pop-down menu of the axis modes available for this occurrence of the axis icon. Only one motion type is allowed for each occurrence of a single icon.

Kinematic Number permits entry of a kinematic library routine number or label identifying the kinematic to be used for ***coordinated*** motion. Kinematic routines are unique to hardware, consult your Indramat sales office for kinematics to drive your hardware.

The drive list box displays setup information for each assigned drive. The type of information depends upon the selected Motion Type. For each drive number the list may contain some of the following: Acceleration, Velocity, Halted, Mode, Trigger 1, Trig 1 Event, Trigger 2, Trig 2 Event, Master number, Slave Ratio, Master Ratio and Change values for each axis.

A horizontal scroll bar on the bottom of the list box allows data to be scrolled into view. Data fields not enabled have "- - - -" in them. Axes can be added, edited, or deleted with the buttons on the bottom.



The **Add**, **Edit** and **Delete** buttons are used respectively to add a drive, modify axis values, and delete an assigned drive. You may also edit a listed drive by double-clicking the list entry. After editing, the dialog box is closed by clicking the Cancel button; all list items are saved as displayed.

Units:

Axis Definition	Position	Velocity	Acceleration
Single Axis Linear	inches	inches/min	inches/s2
Single Axis Linear	mm	mm/min	mm/s2
Single Axis Rotary	degrees(grads)	RPM	rads/s2
Velocity Mode	N/A	RPM	rads/s2
Torque Mode	N/A	N/A	N/A
Coordinated	inches	inches/min	inches/s2
Coordinated	mm	mm/min	mm/s2
Ratioed (master)	*see single axis definitions*		
Ratioed (slaves)	*same as master*		

In Single Axis the drive list may be scrolled to display:

Drive #	Trigger 1
Acceleration	Trigger 1 Event
Velocity	Trigger 2
Halted?	Trigger 2 Event
Mode	

In Velocity Mode the drive list may be scrolled to display:

Drive #	Change
Acceleration	Trigger 1
Velocity	Trigger 1 Event
Halted?	Trigger 2
Mode	Trigger 2 Event

In Ratioed Axes the drive list may be scrolled to display:

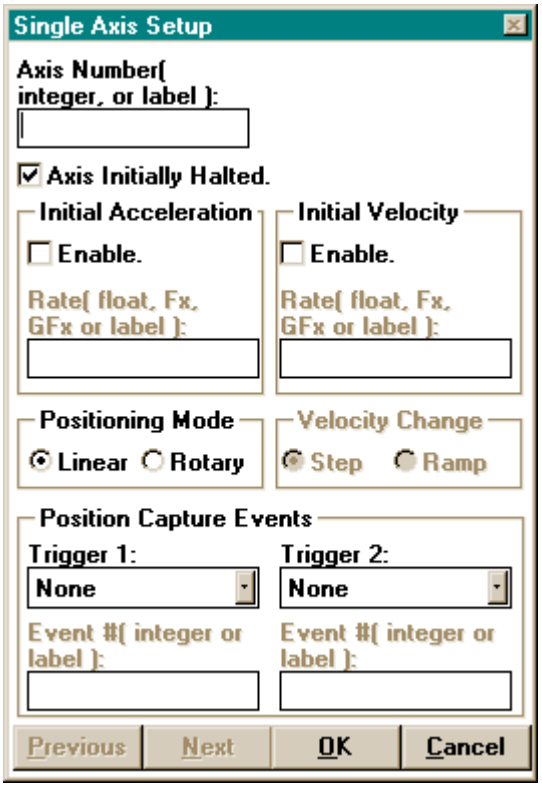
Drive #	Mode
Master #	Trigger 1
Slave Ratio	Trigger 1 Event
Master Ratio	Trigger 2
Halted	Trigger 2 Event

In Torque Mode, the list displays only the drive number. In Coordinated mode, the list displays the drive number and its designation as an x-, y-, z-, roll, pitch or yaw axis.

A pop-up User Defined Labels dialog box is available for the data entry boxes by placing the focus ("I-beam" cursor) in an entry box, then double clicking the left mouse button on an empty area outside of the field.

Selecting **Single Axis** or **Velocity Mode** motion type in the Task Axes Setup dialog box displays another dialog box requiring the entry of an axis to be assigned to the drive. The axis may be entered as an integer constant or equivalent label.

By default, the Axis **Initially Halted** option is checked. This disables the axis at the start of the task; it can be enabled using a Go icon later in the task. Clicking to uncheck the Axis Initially Halted option enables the axis to go at the start of a task.



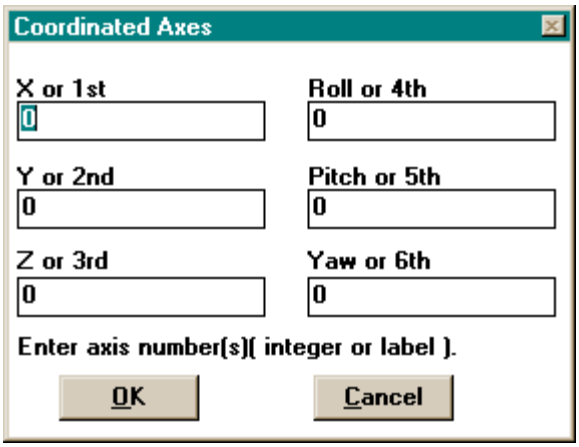
You may enable and enter an **initial acceleration** and **velocity** as a float constant, global float variable(GF1- GF256), program float variable (Fx), or an equivalent label. If the initial accel and velocity are not defined they must be subsequently specified in the program using the ***Accel*** and ***Veloc*** icons. Both Single and Velocity Mode axes may be assigned linear (units) or rotary (degrees) positioning.

Trigger 1, Trigger 2 - enables drive(non-RAC) based position capture for selected drive I/O input, probe 1 or probe 2. Capture can be on 0->1 or 1->0 transition.

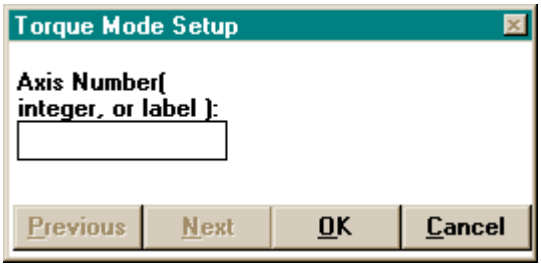
Trig1 Event, Trig 2 Event - event number to execute on position capture occurrence. Event must have been allocated in ***Size*** icon and initialized prior to occurrence. Captured position data can be read in the event's argument field(EVT[x].t) or from the drive parameter utilizing the ***Param*** icon. For more information refer to the ***ProbeEvt*** icon.

Selecting Velocity Mode displays a similar dialog box, however, **Velocity Change** is enabled, permitting selection of Ramp instead of the default Step velocity changes. Ramp velocity changes are based on current acceleration rate, whereas Step changes are made at the maximum rate allowed by the drive and motor.

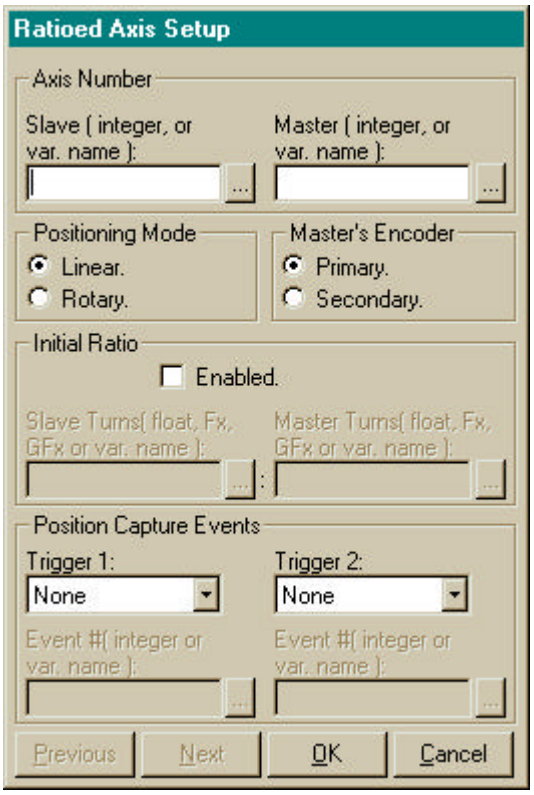
Selecting **Coordinated** motion type in the Task Axes Setup dialog box and clicking the Add button opens a Coordinated Setup dialog box. Axes are assigned by entering an integer or equivalent label for each axis and clicking the OK button. The Coordinated Setup dialog box remains open until it is explicitly closed using the Cancel button. Although you may assign up to 99 axes using one coordinated motion Axis icon, this dialog box is designed to prevent you from assigning multiple axis numbers or labels to a single axis. A pop-up list of labels is available from the data entry box.



Selecting **Torque Mode** allows entering axes in the same manner as coordinated mode; however any number of axes may be assigned up to the maximum number of axes in the system.



Ratioed Axes mode is used to slave one axis to a master (or controlling) axis. The axes may be assigned an **initial ratio** that determines the number of revolutions of the slave to revolutions of the master. If an initial ratio is not defined, this ratio must be set using a Ratio icon. A Ratio icon may also be used in the program to change the ratio. More than one axis may be slaved to a single master. A ratioed axis may be assigned linear or rotary positioning, and may use the drive's probe capability to trigger events.



In CLC Ratio Mode, the slave axis follows the command position of the master axis, if the master is a coordinated motion or CLC cam axis. If the master is any other type of axis, the slave can follow the axis feedback, or an external secondary encoder feedback position.

The axis mode can be switched to single-axis or velocity mode from within the program or for manual mode jogging. It is also possible to “slave off” from the master and position the axis independently. Setting the following ratio to 0 will stop the slave from following the master. A step rate (A-0-0037) allows gradual adjustment of the ratio.

NOTE: *To keep the slave axis synchronized when the master axis is jogging, set Axis Control register bit 4 (Synchronized Jog bit). If the slave axis will never be switched to single-axis mode, this bit should always be mapped high. Ratio Mode Enhancements.*

AxisEvt

The AxisEvt icon is used to enable up to four rotary axis events for a specified axis. A maximum of four events may be enabled by any combination of AxisEvt or Move icons. ***See Chapter 5. Programming Concepts - Repeating Axis (Rotary) Events***.

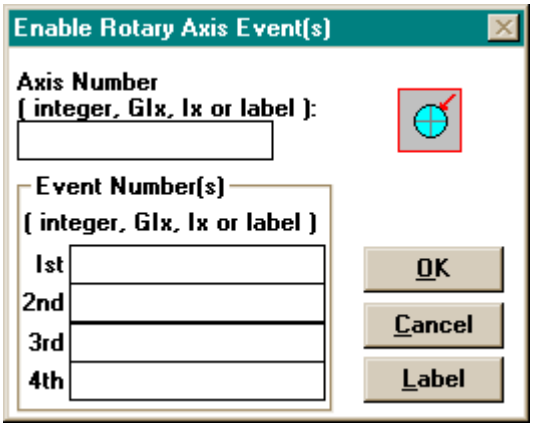
The Axis and Event numbers are specified using integer constants, global integer variable (GIx), program integer variable (Ix), or an equivalent label.

NOTE: *A pop-up User Defined Labels dialog box is available for the data entry boxes by placing the focus ("I" beam cursor) in an entry box, then clicking the cursor on an empty area of the dialog box.*

Each of the four edit fields permit entry of one event (1 - n) that may occur during movement on this path segment. A selected event must exist and configured in the event table prior to run time. The number of events is scaleable in the ***Size*** Icon.

For an event enable to have an effect, the event function must already be entered and setup as a type 5 (Rotary - Repeating Axis Position) event.

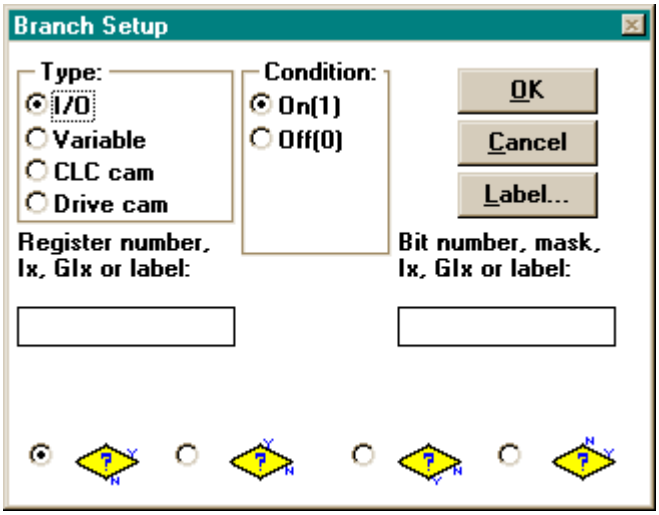
NOTE: *Rotary events cannot be attached to an axis that has a PLS (Programmable Limit Switch) attached to it*.



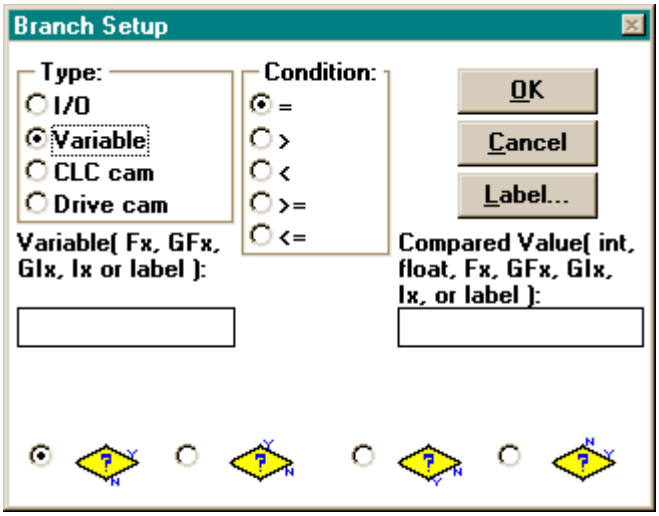
Branch

A Branch icon re-directs the program flow depending upon a true/false logical value.

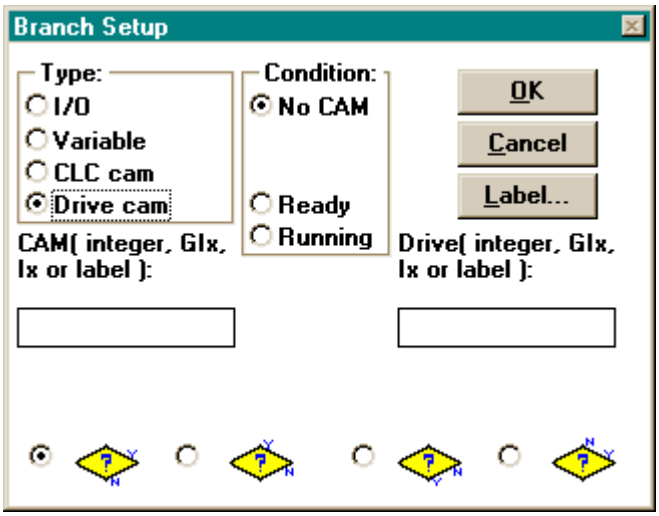
Four radio buttons at the bottom of the dialog box permit selection of one of four Branch icon graphics, each with a different positioning of the branch test outputs.



Selecting an I/O type branch permits testing a bit in a specified I/O register for a logical (true/false) condition. The I/O register for comparison is specified by an integer or an equivalent label. A bit mask permits testing of only selected bits within the specified register to be tested. The bit mask may be entered as an integer or equivalent label. I/O tests are true/false only the equality or inequality relationships are not allowed.



Selecting Variable permits a comparison of the contents of an integer variable, floating point variable or an equivalent label with a compared value (integer or floating point constant, or an equivalent label). User defined labels are accessible through a pop-up dialog box.



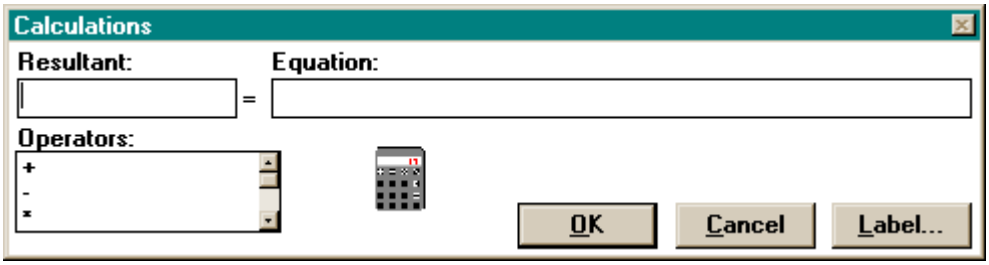
Selecting CLC or Drive CAM permits a specified CAM check for one of three conditions. The branch will return a yes or no value if there is No CAM or if the Cam is Ready or Running.

Calc

The Calc icon can be used to initialize points and zones, and setup events. By using Calc icons in this way, points, zones and events reside within the programs, thereby eliminating the need to specify them after downloading the program to a CLC. The **Resultant** box specifies the element of the point, event or zone tables, or an equivalent label. The **Equation** box contains the expression to be evaluated.

Example:
Absolute point definition ABS[1]={10,10,0,0,50,50,50,90,2,0,0,0,0,0,0,0}

Distance-based event setup EVT[1]={1,3,0,95,vac_on,executing event}



For descriptions of point, zone and event definitions, ***refer to the information on point, zone and event tables in Chapter 5. Programming Concepts***

Calculations

The Calc icon also permits placing a calculation into the CLC program flow. The **Resultant** box specifies destination of the calculation and must be a variable (Fx, GFx, GIx, Ix), point table element (ABS[x].{ x, y, z, b, s, a, d, j, e1, e2, e3, e4, r, p, ya }, REL[x].{ }), event table element(EVT[x].{ s, t, r, a, f, m}), zone table element(ZONE[x].{ s, x^1, y^1, z^1, x^2, y^2, z^2 }), PLS[1].{a, o, r, t, on1-on16, off1-off16} or equivalent label. *A pop-up User Defined Labels dialog box is available.* One level of indirection can be used in the resultant(i.e. F[I7] or ABS[I3].x). The index used for the indirection must be an integer variable, global or program(GIx, Ix).

Coordinated Motion Variables

The following variables are used for coordinated motion:

ABS[x].el or **REL[x].el** These variables reference the absolute (ABS) or relative (REL) point table, and they can define the elbow direction (for SCARA kinematics) OR mask movement on one or more axes (for Cartesian kinematics--#1). The value for these variables is entered as an integer, representing a 4-bit binary value, which is defined as follows:

Bit	Property	Integer	Kinematics
0	Elbow up/down (up=1; down=0)	1 (2^0)	SCARA **only**[1]
1	Mask movement on Axis x (on=1; off=0)	2 (2^1)	Cartesian (#1) **only**[2]
2	Mask movement on Axis y (on=1; off=0)	4 (2^2)	Cartesian (#1) **only**[2]
3	Mask movement on Axis z (on=1; off=0)	8 (2^3)	Cartesian (#1) **only**[2]
[1]The elbow up/down property is ignored by any kinematics besides SCARA. [2]The axis masking property is ignored by any kinematics besides #1.			

ZONE[x].s This variable defines zone status as active or inactive for one or more tasks (or all tasks), and it can define the zone boundaries by the coordinates of opposite corners of a cube in space. If coordinates are defined, they are relative to the origin of each robot (e.g. if the robot in Task A has a different origin than the robot in Task B, the same zone must be defined with different coordinate values for each task, relative to each robot's origin).

The value for the status is entered as an integer, representing a five-bit binary value, which is defined as follows:

Bit	Zone Status	Integer
0	All tasks (active=1; inactive=0)	1 (2^0)
1	Task A (active=1; inactive=0)	2 (2^1)
2	Task B (active=1; inactive=0)	4 (2^2)
3	Task C (active=1; inactive=0)	8 (2^3)
4	Task D (active=1; inactive=0)	16 (2^4)

Enter the expression to be evaluated in the **Equation** box. Operators may be selected from the Operators scrolling menu or entered using the keyboard. Symbolic labels that equate to a constant may be used in equations. The equation is evaluated from left to right. Parentheses can be used to control the order of evaluation.

NOTE: *The Operators menu includes type conversion operators for floating and integer data types. These operators should be placed immediately to the left of the operand (i.e., as the equation is parsed from left to right, the compiler first obtains the value, then performs the conversion.).* ***See Chapter 5. Programming Concepts - Mathematical and Logical Operators.***

Although an equation may use a combination of floating point, integer variables and constants, be sure that the data types used for a single operation match. Data type mismatches are especially easy to overlook when using symbolic label names that do not implicitly identify the data type. Use the "float()" and "int()" conversion operators to force the proper data types.

Examples using data structure elements:

ABS[1].x = ABS[2].x

REL[5].z = 50.9 + F4

ABS[2].el = 0 (elbow down)

REL[2].el = 1 (elbow up)

ABS[1].el = 12 (mask movement on Axes y and z)

REL[1].el = 2 (mask movement on Axis x)

EVT[I1].f = sam1

I3 = EVT[1].s

PLS[1].r = 100

EVT[1].m = "SAVE THE WORLD"

ZONE[I1].b = 123.0

In addition to equations, the Calc box can be used to transfer data between like data structures and to initialize data structures.

Examples of data transfer between like structures:

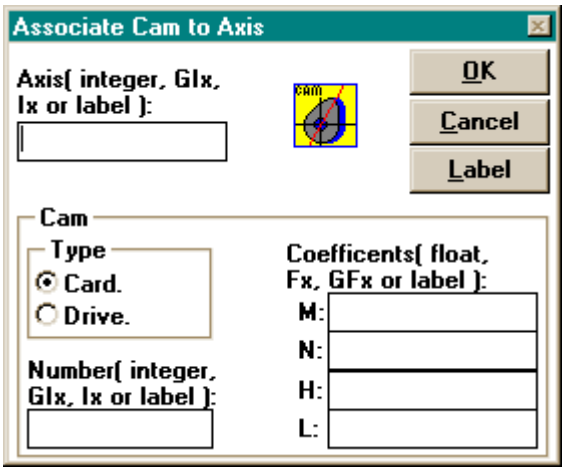
ABS[1] = ABS[2]
REL[1] = REL[2]
EVT[1] = EVT[2]
ZONE[1] = ZONE[2]
ABS[1] = REL[2]
ABS[I1] = ABS[GI2]
ABS[sam] = ABS[read]

Examples of data structure initialization:

ABS[1] = {10.1,11.2,12.3, 1.0,55,66,77,88,1,2,3,4,23.0,24.0,25.0, 1 }
REL[1] = {10.1,11.2,12.3, 1.0,55,66,77,88,1,2,3,4,23.0,24.0,25.0, 1 }
EVT[1] = { 1, 2, 3, 10.1,sam, "HELLO WORLD" }
EVT[1] = { 1, 2, 3, 10.1,0, "HELLO WORLD" }
EVT[1] = { 1, 2, 3, 10.1,0, "" }
EVT[1] = { 1, 2, 3, 10.1,NONE, "" }
ZONE[1] = {1, 10.1,11.2,12.3,13.4,14.5, 15.6 }

Cam

The Cam icon is used to associate a cam to a slave axis and to supply coefficients for using the cam. To work properly, the axis must be configured as an ELS slave and synchronized to the master. The Cam can be drive or Card resident.



Axis: Specifies which axis is associated with the cam. The entry may be an integer, global integer variable (GIx), program integer variable (Ix) or an equivalent label.

Cam Number: Specifies which Cam table to use. The entry may be an integer, global integer variable (GIx), program integer variable (Ix) or an equivalent label (Range 1 to 8 for CLC cams and 1 to 2 for DDS2.2 cams.)

Coefficients: M, N, H and L may be entered as floats, global float variables (GFx), program float variables (Fx), or equivalent labels (card cams only).

For more information see Appendix C. ELS Configuration.

CamAdj

This icon selects which phase adjust to perform and starts the phase adjust. There are two phase offset values for a cam axis: a master phase adjust, and a slave phase adjust. The master phase adjust shifts the position in the cam table relative to the master position. The slave phase adjust shifts the position of the slave axis. Since it is not related to the shape of the cam, the slave phase adjust is not multiplied by any of the cam factors.

The Master Phase adjust adds the phase adjust value to the sampled ELS master position before the slave position is referenced in the cam table. For CLC Cams, the Master Phase Adjust writes to axis parameter A-0-0151. For drive based cams, the Master Phase Adjust writes to drive parameter P-0-0061.

The Slave Phase adjusts the phase adjust value to the output position of the cam table. For CLC Cams, the Slave Phase Adjust writes to axis parameter A-0-0161. For drive based cams, the Slave Phase Adjust writes to axis parameter A-0-0151.

Type selects the method of adjustment:

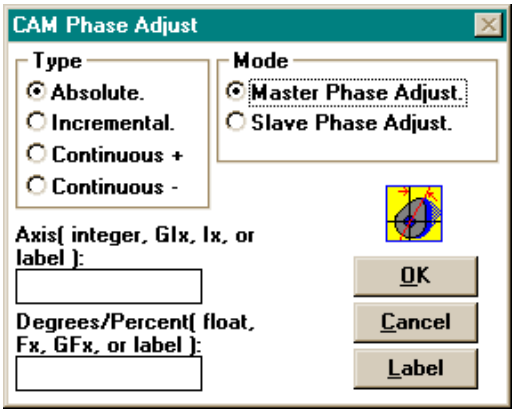
If **absolute**, the *degrees or percent* edit field is the new offset.

If **incremental**, the *degrees* edit field is added to the current offset. If in phase mode and sum exceeds 360, it rolls over. If in velocity mode the sum is limited to -100 or +300 percent.

If **continuous +** or **continuous -** (phase sync mode only), a velocity dependent amount is added to the degree offset.

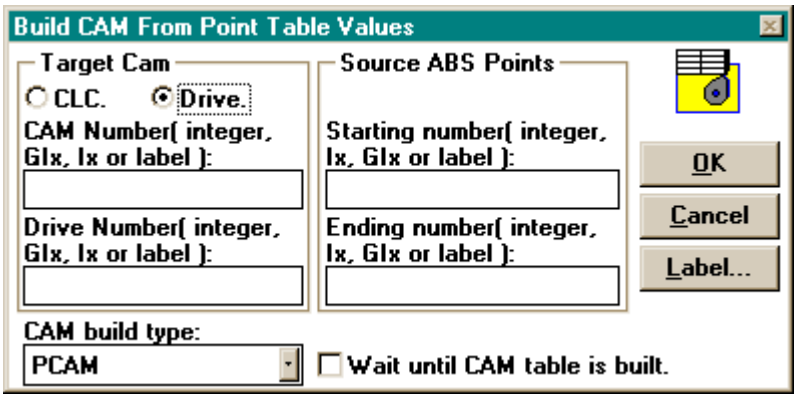
The CLC can perform only one phase adjust at a time. If two different phase adjust icons are used the second one will have no effect until the previous phase adjust is complete. Bit 4 in the axis status register is set to (0) when a phase offset is in progress, and (1) if the phase offset is complete.

Type:	Absolute Incremental (relative) Continuous plus Continuous minus
Mode:	Master phase adjust (Mph in equation) Slave phase adjust (Sph in equation)
Axis:	Axis number
Deg/Per:	Phase adjust target value



CamBuild

The CLC can build a cam on-line based on a set of input positions. Program instructions use the CLC's ABS point table and an internal utility to build an internal CLC Cam which is stored on the Card or Drive as the Cam number indicated. ***For more information see Appendix C. ELS Configuration*.**



Cam Number - indicates the cam table to build. The entry may be an integer, global integer variable (GIx), program integer variable (Ix) or an equivalent label (range 1 to 8 for CLC cams and 1 to 2 for DDS2.2 cams). The cam cannot be in use when this icon is executed.

Drive Number - (drive cams only) indicates which drive the Cam is stored in. The entry may be an integer, global integer variable (GIx), program integer variable (Ix) or an equivalent label.

Cam build type (PCAM, VCAM, ACAM or Spline) - The X elements of the point table are the master positions, and the Y elements are the corresponding slave positions. The **PCAM** build type accepts input in the form of a table of target **positions**. The **VCAM** build type accepts a **velocity** profile as input and outputs a normalized profile. **ACAM** build type accepts an **acceleration** profile as input and outputs a normalized profile. When **Spline** is selected the CLC builds the cam by connecting the points with third order splines. A minimum of 5 and maximum of 200 user defined points are allowed.

ABS Starting Point - is the starting ABS point number used for cam generation. The entry may be an integer, global integer variable(GIx), program integer variable(Ix), or an equivalent label. The number must be in the range of points defined in the ***Size*** Icon.

ABS Ending Point - is the ending ABS point number used for cam generation. The entry may be an integer, global integer variable(GIx), program integer variable(Ix), or an equivalent label. The number must be in the range of points defined in the ***Size*** icon.

The ABS point elements may be changed from within the program using CALC statements. Changes in the point table do not affect the cam until the CAM BUILD icon is executed. It is necessary to size the point table at compile-time to allow enough points for the profiles that will be needed. The x value of the first point must be 0, and the x value of the last point must be 360.

The cam generation may take one second or longer. The 'wait' checkbox can be cleared to exit this icon immediately and keep executing instructions while the cam is being built. The branch icon can be used to check if a cam is ready for activation. If the 'wait' checkbox is checked, the program flow will be stopped in this icon until the cam is ready for activation.

The CAM BUILD icon can be used to store a cam to an inactive location on the CLC or drive. After the cam has been built, the CAM icon can select it for an axis.

Because the cam is stored in nonvolatile memory on the CLC or the drive, it is not necessary to execute this command each time through the program. A flag variable can be set and checked the next time through the program to avoid long delays when starting the program. For on-line changes, a register bit or variable should be checked each time through the program loop to avoid continually generating the cam, which consumes CLC resources and can slow down the program.

CLC cams enter the CLC as text files in CSV format, the kind most spreadsheets generate. Drive cams are a single column text file indicating the slave position.

If the CSV file for CLC cams contains less than 1024 points, an algorithm within the CLC fills in the missing points. As a rule, a CSV cam file should contain at least 200 points, anything less than that does not sufficiently define the cam and unexpected results may occur. Drive cams must contain exactly 1024 points.

Errors at runtime

The selected cam is currently active for any axis.

The point range exceeds the bounds of the point table.

Less than two points are defined.

The cam number is not valid (out of range or drive is not configured).

An error occurred when sending the cam to the drive.

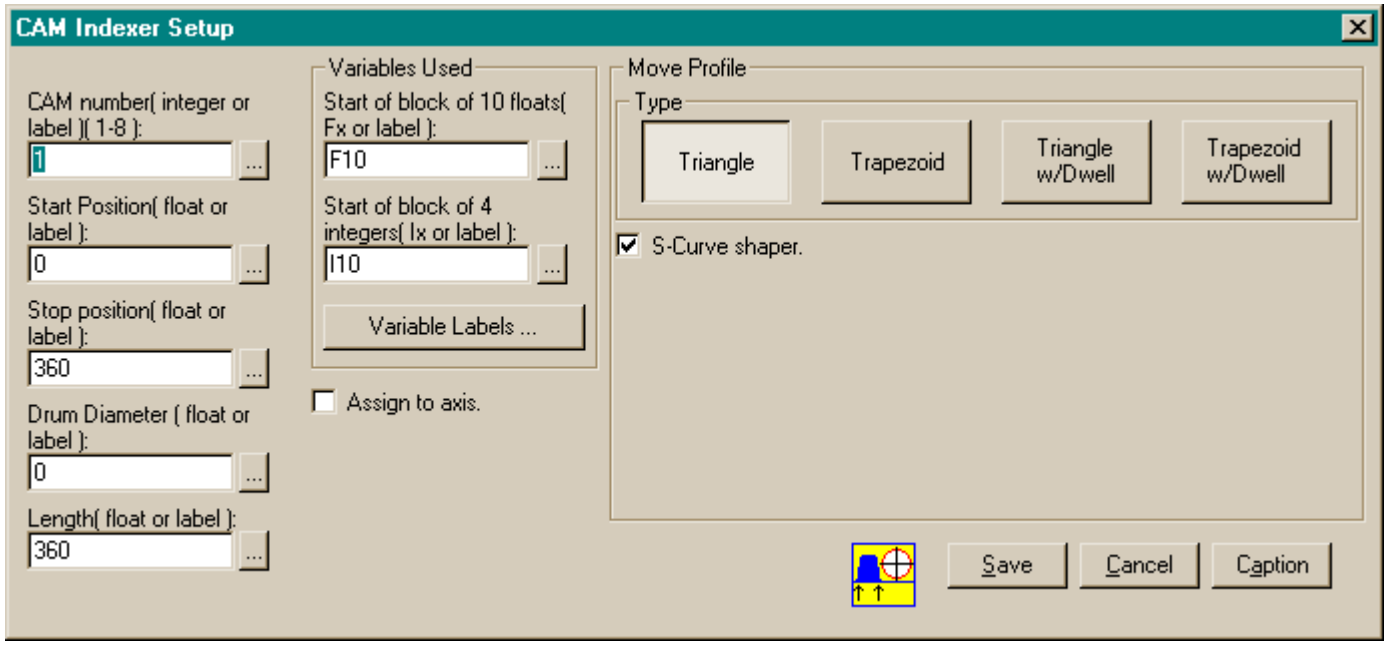
When using PCAM option, and the first x position isn't 0 and the last x position isn't 360.

The x position exceeds the modulo of the master.

Cam Indexer

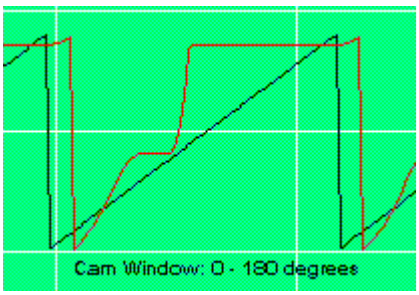
NOTE: *When using Indexer Cams, the SERCOS cycle time (C-1-0099) needs to be increased to 4ms or more to allow time for the equation calculations.*

Index cams are CLC cams that use equations to compute a position, as opposed to a normal cam, which uses a point table. They operate in real-time which allows their start and end positions to be freely changed within the next index cycle. This makes them ideally suited for high speed film feed applications where the seal time must be held constant over line speed.



CAM number: Identifies the CAM to use (1-8)

Start Position (Ps): Defines the virtual or real master position (in degrees) at which the cam index profile starts. **Stop (Pe) Position:** Defines the virtual or real master position at which the cam index profile stops.



These two parameters define the cam's cycle time, **Tc**.

$$\mathbf{Tc} = \text{ABS}(\mathbf{Pe}\text{-}\mathbf{Ps})/\text{Vm}$$

where ABS is the absolute value of the difference and **Vm** is the velocity of the master in degrees/sec.

Length (D): Defines the distance (in EE units) the slave axis moves during an index cycle. The velocity profile developed depends on the start/stop positions, profile type, index length and master speed.

Drum Diameter (Dia): Allows for automatic scaling of the index distance into degrees. The following equation is applied:

$$\text{Angle} = (\mathbf{D}/\mathbf{Dia})(360/\pi)$$

Internal cam units (degrees) are computed automatically through this parameter.

NOTE*: When the Drum Diameter is set to zero, its default value is 360°/rev. The benefit of this is that internal 64 bit math can be used to minimize rounding errors.*

Variables Used: Defines the starting point for a block of variables that will be assigned values for the CAM Indexer function. The value of these variables are set according to the data entered in this dialog box.

10 float variables (F(n) , F(n+1), F(n+2)… (Fn+9)

CAMI_01_ PRO_LENGTH	;CAM 1 Indexer, profile length
CAMI_01_ PRO_START	;CAM 1 Indexer, start position of profile
CAMI_01_ PRO_STOP	;CAM 1 Indexer, stop position of profile
CAMI_01_ DIAMETER	;CAM 1 Indexer, drum diameter
CAMI_01_ EVENT_TIME	;CAM 1 Indexer, time period before end of move
CAMI_01_ ACCEL	;CAM 1 Indexer, percent cycle for acceleration
CAMI_01_ DECEL	;CAM 1 Indexer, percent cycle for deceleration
CAMI_01_ DWELL	;CAM 1 Indexer, time period of dwell
CAMI_01_ PRE_DWELL	;CAM 1 Indexer, percent cycle before dwell
CAMI_01_ RESERVE_F1	;CAM 1 Indexer, reserve float 1

4 integer variables (I(n) , I(n+1)… I(n+3)

CAMI_01_ FLAG1	;CAM 1 Indexer, flag word 1
CAMI_01_ PRO_TYPE	;CAM 1 Indexer, profile type
CAMI_01_RESERVE_I2	;CAM 1 Indexer, reverse integer 2
CAMI_01_ RESERVE_I1	;CAM 1 Indexer, reverse integer 1

Assign to Axis: Optionally allows the Indexer cam to be assigned an axis. The axis must have been declared as a card cam in the ELS icon. A cam can also be assigned, or reassigned using the CAM icon.

Enable Registration Correction: This enables the auto-registration function for the indexed cam. ***For more information refer to the Registration Icon.***

The following types of Move Profiles are available:

Triangle	Step Acc/Dec of 50%
Triangle (S-Curve)	Ramp Acc/Dec of 50%
Trapezoid (Fixed)	Step Acc/Dec of 33%
Trapezoid	Step Acc(x%)/Dec(y%), x+y≤100%
Trapezoid (S-Curve shaper)	Ramp Acc(x%)/Dec(y%), x+y≤100%
Triangle w/Dwell	Step Acc/Dec, center dwell
Triangle w/Dwell (S-Curve)	Ramp Acc/Dec, center dwell
Trapezoid w/Dwell (S-Curve)	Ramp Acc/Dec, center dwell
Trapezoid w/Dwell	Step Acc/Dec, center dwell
Trapezoid w/Dwell	Step Acc(x%)/Dec(y%) x+y≤100%, Sw% dwell

Cams with ramped acceleration (S-shaped velocity profile) are defined as third order polynomials, those with stepped acceleration (ramp velocity profile) as second order polynomials.

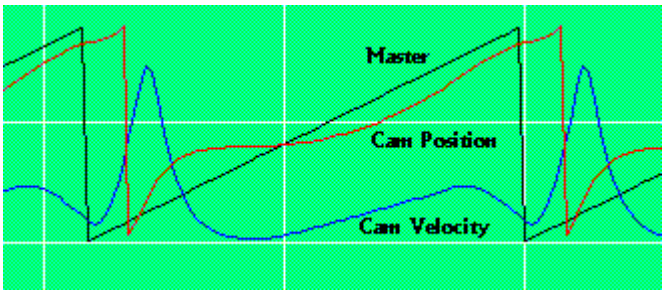
The Triangle profile without the S-Curve shaper has the simplest equation, thus it takes the least amount time to execute, but has maximum jerk. Selecting the S-Curve shaper takes longer but minimizes jerk.

The Trapezoid profile provides cam shaping through an accel./decel. percent setting but takes the longest to execute due to equation complexity.

Cams with dwells are slightly more complex than those without dwells and so require slightly more time to execute than those without dwells.

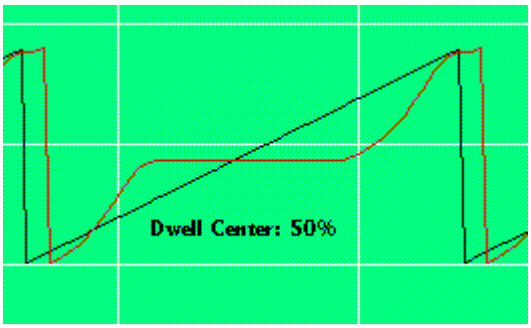
Cam Acceleration/Deceleration(Acc/Dec) - Some cam types (see table) use these parameters to define the percent of their cycle time devoted to acceleration and deceleration.

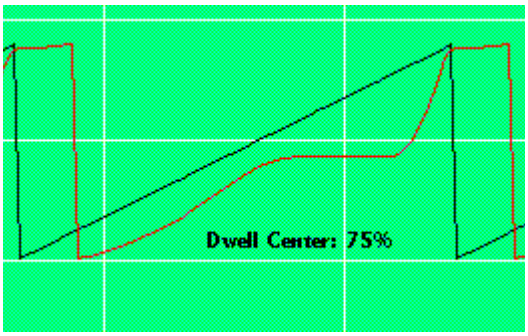
Example: A Trapezoid cam profile with an acceleration setting to 20% and deceleration setting to 50% would step accelerate for 20% of its cycle time, slew for 30% and decelerate for 50%.



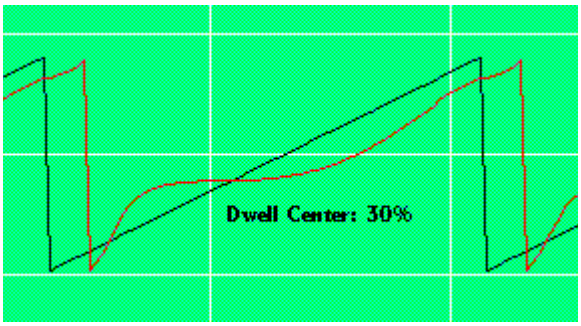
Cam Dwell (Cw) - Some cams (see table) use this parameter to determine the center of the dwell. This is defined as a percent which is limited to 25% to 75% of the cam cycle.

Example1: Cw = 50% Ps = 0 Pe=360 then dwell center
Pw = (360-0)*0.5 + 0 = 180 deg.





Example2: Cw = 30% Ps=90 Pe=270 then dwell center
Pw = (270-90)*0.3+90 = 144 deg.



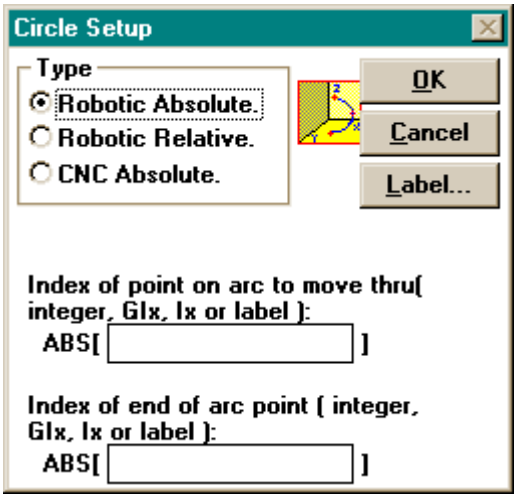
Cam Dwell (Tw) - Some cams use this parameter to determine the amount of dwell within their cycle. A dwell is a period of time during which the cam is stopped. Tw is limited by the following relationship:

Max(Tw) = 2.0*Min[0.40*(Pe-Pw)/Vm, 0.40*(Pe-Pw)/Vm]

where Pw (deg) is the dwell center angle and Vm (deg/sec) is master speed.

Circle

The Circle icon is used for multi-axis coordinated circular interpolation, moving in a circular arc from one point in 3D space to a second point in 3D space. The plane of the arc is two dimensional and the plane may have any orientation in three dimensional space. Placing a Circle icon on a Task or Subroutine workspace opens a Coordinated Circle Setup dialog box. The Go Icon is not required with Circle function. When executed, the motion will immediately be sent to the path planner. Stepping to the next icon in the program will take place immediately.



Robotic Move

Move in a circular arc from the current point in space past the “thru” point to the end point. The plane of the arc is two dimensional and may have any orientation in three dimensional space.

A circular move may be Absolute or Relative and is defined by three points on the circle. These points must have been previously defined in the absolute and/or relative point tables. Points are identified by a point table ID number or equivalent label. Indirect point table entries may be used by specifying an integer variable, global or program(GIx, Ix) or equivalent label.

An absolute circular move begins from the endpoint of the previous segment (or current position if the system is halted), moves through an absolute point, and terminates at another absolute point.

A relative circular move is similar; however, the intermediate and endpoints are defined as relative offsets from the starting point. The relative circular move begins at the endpoint of the previous segment (or current position if the system is halted), moves through the first relative offset, and terminates at the second relative offset.

CNC Move (Reserved for future use)

Move in a circular arc in the defined plane. Start of the arc is the current position. Center of the arc is C(I,J) in the defined plane. End of the arc is the intersection of the arc and a line from the center to end point E(X,Y), or the number of degrees specified from the start.

The center point, end point, degrees, direction, feed rate, and defined plane are defined in the ABS point structure as follows:

X	Y….	Z….....	S	R	P	W	El
I	J…..	DIR…	F	DEG	X	Y	1=x-y plane
I	J…..	DIR…	F	DEG	X	Y	2=z-y plane
I	J…..	DIR…	F	DEG	X	Y	3= y-z plane

The arcs radius is the distance from start to center.

The arcs direction is:

DIR > 0 - Clockwise.

DIR <= 0 - Counterclockwise.

The end of the arc is:

DEG=0 -The intersection of the arc and a line from the center to end point.

DEG>0 - The number of degrees from a line through start and center points.

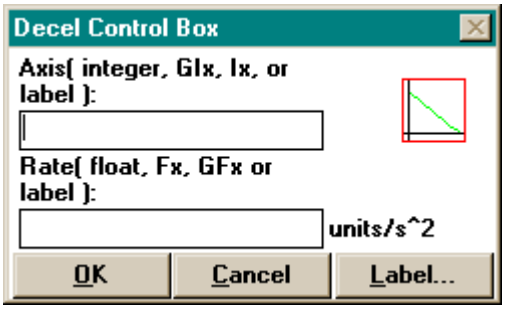
Decel

The Decel icon is used to specify or change the deceleration rate of a virtual axis(0) in ELS mode or an axis in velocity mode.

Deceleration rate - specifies the deceleration rate in units per second per second. The entry may be a float, global float variable(GF1- GF256), program float variable(Fx), or an equivalent label.

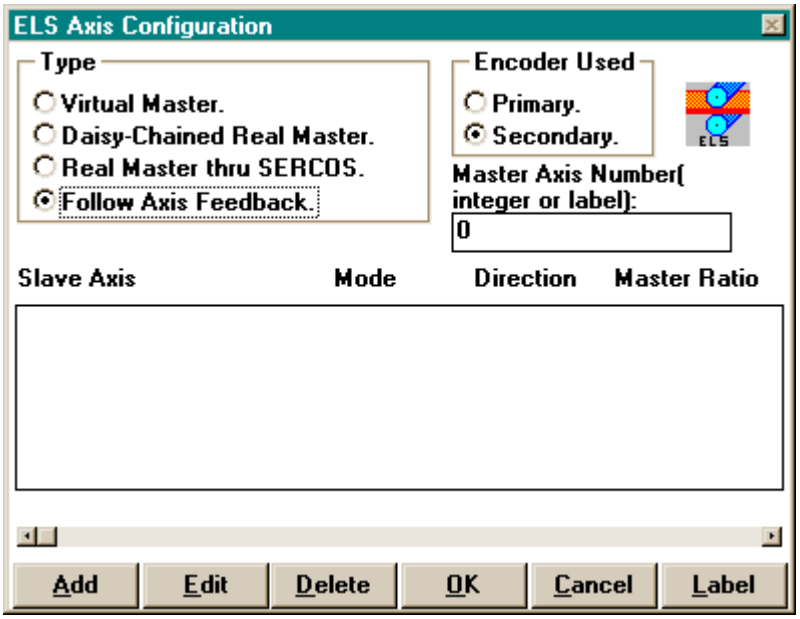
A pop-up User Defined Labels dialog box is available for the data entry boxes by placing the focus ("I" beam cursor) in an entry box, then clicking the cursor on an empty area of the dialog box.

Units: units/s^2



ELS

The ELS icon is used in place of the Axis icon to associate and configure slave axes in a CLC system that uses the ELS (Electronic Line Shaft) capability. ***For more information see Appendix C. ELS Configuration***. Only one task in a program can be used in the ELS mode.



Virtual Master uses the CLC's ELS virtual master (axis 0) to provide master position data (units = degrees). The Virtual Master is controlled in the user program with the accel, decel, velocity, go and stop icons and it sends this position data to all drives; therefore all drives on the CLC SERCOS communication loop have equal synchronization delays. Drives are synchronized within the single SERCOS cycle to axis 0, the virtual axis.

Real Master through Hardware *(Daisy-Chained):* configures DDS-2 drive systems that use DFF cards. The DFF cards are interconnected using an independent cabling system and are synchronized to the drive containing the master card. Although the DFF cards provide the synchronization between drives, DFF system drives must still be configured using CLC Axis Configuration. **Note:** *Daisy-Chained Real Masters are no longer supported in GPS5 and DIAX03 drives..*

Real Master through SERCOS reads position data from the master drive's external encoder, then sends the position data to the slaves through the SERCOS communication system. This configuration is used when only one drive in the system contains a DFF master encoder card and the master position is transmitted to the other slave axes through SERCOS. Slaves are synchronized, but delayed one SERCOS cycle to read the master, then one cycle to transmit to the slaves. The master axis must be defined and configured using an Axis icon.

Follow Axis Feedback reads position data from an encoder through SERCOS communication, then sends position information to the slave through SERCOS. This configuration is used when a real axis on the SERCOS ring will be the ELS master. The axis must be defined in the user program as single axis, rotary mode. Slaves are synchronized but may be up to two SERCOS cycles delayed from the master. Encoder Used permits selection of the drive's primary or secondary encoder. The master axis must be defined and configured using an Axis icon.

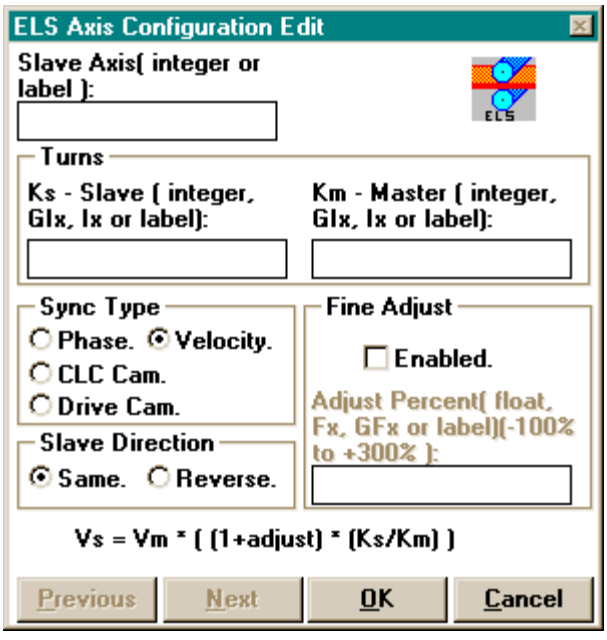
The drive list displays the slave axis:

Mode - velocity, phase or cam
Direction - positive or negative with respect to the master
Master Ratio - an integer value for the master used to compute the master/slave ratio
Slave Ratio - integer value for the slave used to compute the master/slave ratio
Adjust - the percentage difference use to compute the slave velocity or position

Clicking Add, or selecting a listed drive then Edit (or double clicking the list item) opens an ELS Axis Configuration Edit dialog box.

A list entry can be deleted by clicking and highlighting the entry, then clicking the Delete button.

Clicking OK in the ELS Axis Configuration dialog box saves the changes, clicking Cancel abandons any changes and returns to the Visual Motion workspace.



The ratio of the slave to the master is set by entering integer **Turn** values for both slave and master. The CLC calculates the ratio insuring that the highest

accuracy possible within the system is maintained. The Turns ratio determines the modulo of the slave axis: modulo = 360 * Ks/Km (degrees).

Units:

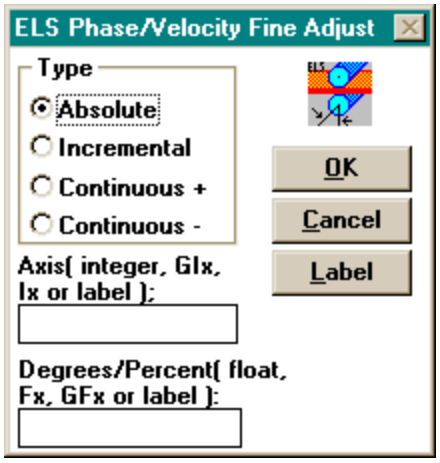
Axis Definition	Position	Velocity	Acceleration
Virtual Master	degrees	RPM	rads/s2
Real Master Encoder	degrees (displayed only)	RPM (displayed only)	N/A
Follow Axis Feedback	degrees	RPM	rads/s2
Slave - Phase	degrees (displayed only)	RPM (displayed only)	N/A
Slave - Velocity	N/A	RPM (displayed only)	N/A
Slave - Cam	degrees	RPM (displayed only)	N/A

Drives are synchronized in terms of velocity, phase or cam position. If enabled, an adjustment factor may be entered (multiplied constant for velocity, additive constant for phase: ***see the ELSAdj icon or the Cam icon for more information***). The slave drive may have normal or reversed direction with respect to the master.

Clicking the OK button places the new or edited configuration into the drive list. Multiple drives may be added or edited from the same Configuration dialog box. Clicking Cancel returns to the ELS Axis Configuration dialog box. Additions and edits entered using the OK button are displayed in the drive list.

ELSAdj

The ELSAdj icon is used to adjust the velocity or phase of an ELS configured axis to compensate for mechanical variations between master and slave axes. The axis may be specified by an integer constant, variable, global variable or an equivalent label. A pop-up User Defined Labels dialog box is available from both data entry boxes.



Type selects the method of adjustment:

If **absolute**, the *degrees or percent* edit field is the new offset.

If **incremental**, the *degrees or percent* edit field is added to the current offset. If in phase mode and sum exceeds 360, it rolls over. If in velocity mode the sum is limited to -100 or +300 percent.

If **continuous +** or **continuous -** (phase sync mode only), a velocity dependent amount is added to the degree offset.

For phase synchronization the resulting slave phase is:

$Ø_S = (Ø_m * (K_S/K_m)) + adjust$

For velocity synchronization the resulting slave velocity is:

$V_S = V_m * ((1 + adjust) * (K_S/K_m)$

Where:

$Ø_m$ = master axis phase

V_m = master axis velocity

K_S = slave axis master/slave ratio turns value

K_m = master axis master/slave ratio turns value

$adjust$ = a value in the range of -100% to +300% for ratio, or 0° to +360° for phase.

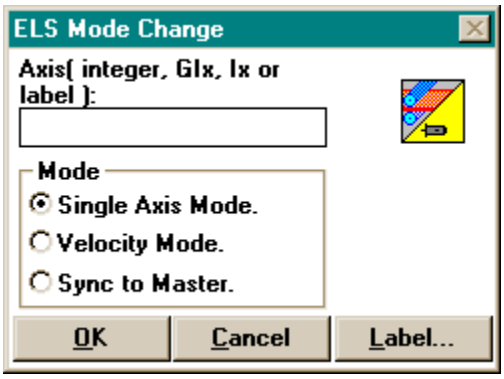
Axis - indicates the axis to be adjusted. The entry may be an integer, global integer variable(GI1- GI256), program integer variable(Ix), or an equivalent label.

Degrees(phase) or percent(velocity) - When enabled this allows adding a fine offset. This entry may be an float, global float variable(GF1- GF256), program float variable(Fx), or an equivalent label. Valid adjustment ranges are 0 to 360 degrees or -100% to +300%.

ELSMode

The ELSMode icon is used to switch between ELS, Single Axis and Velocity Modes. Once switched into Single axis mode, the axis may be positioned independent of any ELS master/slave relationship (i.e., jogged into position), then returned to Sync to Master restoring the relation.

The ELSMode icon can also be used to switch an axis that is configured for Single Axis Mode into Velocity Mode. Axis parameter **A-0-0180** must be set to 36 to put command velocity into the cyclic data. Axis parameter **A-0-0004**, **bit 7** must be set to 1 to enable acceleration. The Sync to Master mode is then ignored.



The axis is specified by an integer constant, variable, global variable or an equivalent label. Axes must be configured as ELS axes. A pop-up User Defined Labels dialog box is available from both data entry boxes. boxes. Entering “-1” in the axis box will send the mode change command to all ELS axes defined in the Task where the command is issued.

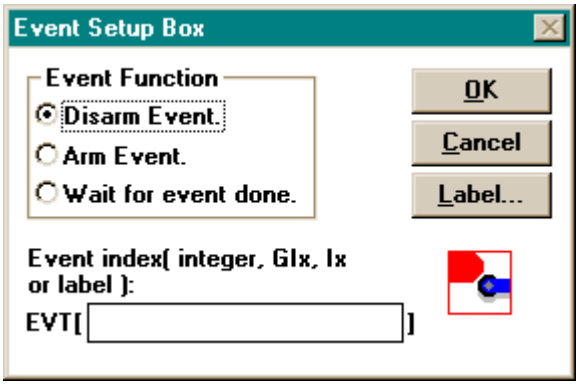
When an axis that is configured for ELS Phase or Drive Cam mode is switched into sync mode, a relative phase offset is automatically initialized between the slave and the master. The drive does not move into absolute synchronization with the master.

Care should be taken when switching an axis into synchronization with a moving master. With Diax02 drives and Diax03 Version 3 drives, the slave does not ramp to the master speed. It will try to accelerate at full torque. Diax03 version 4 drives have the “ramp up and lock on” feature that assures smooth acceleration when synchronizing to a moving master.

When an axis is switched to single axis mode while the master is moving, it does not decel to a stop. It will stop at the last valid position command. To switch to single axis mode from following a moving master, first switch to velocity mode. The slave will then continue moving at the last sampled master velocity (even in phase sync) and then can be ramped down to a stop using the decel and stop icons.

Event

The Event icon is used to control the way that the system handles events; and may be used to control program flow by suspending a task's program execution pending the completion of an event. For the event icon to have an effect upon system operation, an event must have been previously entered in the CLC event table. Placing an Event icon on a task or subroutine workspace automatically opens an Event dialog box.



The event is specified by entering an event ID number indicating a valid event in CLC's event table. The event may be entered as an integer constant, variable (Ix), global variable (GIx), or an equivalent label.

An event icon may have one of three effects:

Disarm Event de-activates the specified event. If the event has already been made inactive, the icon has no effect.

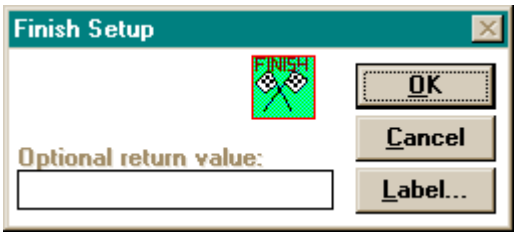
Arm Event is valid only for timed events. An Arm Event icon starts the specified event's timer. If the event is currently active, the icon has no effect.

Wait for event done suspends the execution of program flow until the specified event has completed.

For more information see Chapter 5. Programming Concepts.

Finish

Each program task, subroutine and event function must end with a single Finish icon. Subroutines can return an optional single argument to the calling function. The return argument may be a constant or a variable. A return argument is a convenient way to get position when utilizing a common subroutine from more than one task. The return argument only works when the ***Sub*** icon requests it.

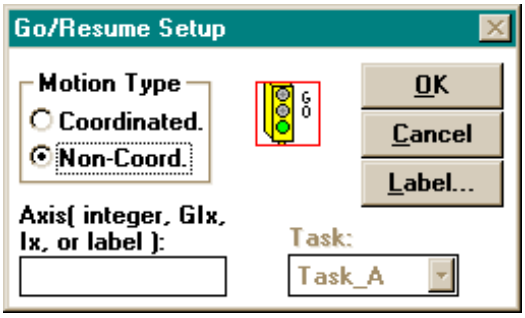


Go

The Go icon is used to enable one or all non-coordinated axes used in any task. It also enables the associated position, velocity or servo loops. It should be placed before an associated Move icon. The Go icon can also be used to resume multi-axis coordinated motion that has been stopped.

Placing a Go icon automatically opens the Go Setup dialog box. The **Motion Type** radio button specifies the type of motion to start. Choosing Single Axis Motion Type requires an entry in the **Axis** data entry box specifying the CLC axis to start. The axis is specified by a valid integer constant, variable, global variable or an equivalent label. Specifying "-1" as the single axis enables all the single axes assigned to the current task in the Axis icon. Specifying "0" starts the virtual master of an ELS system.

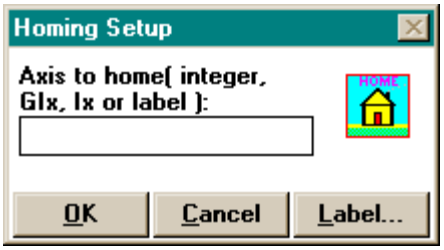
Choosing Coordinated Motion Type to resume motion requires selection of one of the four CLC tasks from the pull-down **Task** pick list. The coordinated motion must have been halted by a coordinated stop. Aborted coordinated motion should not be simply resumed.



A pop-up dialog box of User Defined Labels is available.

Home

The Home icon commands the CLC to send an axis home signal to the specified drive. This is a single-axis non-coordinated motion command. Placing a Home icon on a task or subroutine workspace automatically opens the Homing Setup dialog box.



The axis to home may be specified by a valid integer constant, variable (Ix), global variable (GIx), or an equivalent label. A pop-up dialog box of User Defined Labels is available.

Once commanded to home, the drive homes the axis to the home position without further intervention from the CLC. Any errors in the drive's homing operation are reported to the CLC. The user program waits in this icon until the homing sequence is complete.

The Home icon uses the internal homing capability of the intelligent DDS drive to perform the homing operation. For homing to occur, the homing parameters in the specified drive must have been setup prior to executing the Home icon. The homing parameters may be set using the Homing Setup dialog box accessed through the Drives menu item under *S*etup. **See the *Drive Reference* menu selection under *Parameters* in the *CLC Drive Parameter Editor*.**. Refer to the Drive manual for more information about homing and the homing parameters.

The Home icon cannot be used with multi-turn feedback. To reference multi-turn feedback ***see Appendix D. CLC Drive Parameter Editor.***

I/O

The I/O icon is used to control the state of I/O register bits. Placing an I/O icon on a task or subroutine workspace automatically opens an I/O Setup dialog box. Entering a number as a register ID code, or an equivalent label, in the I/O Register data entry box selects the target I/O register.

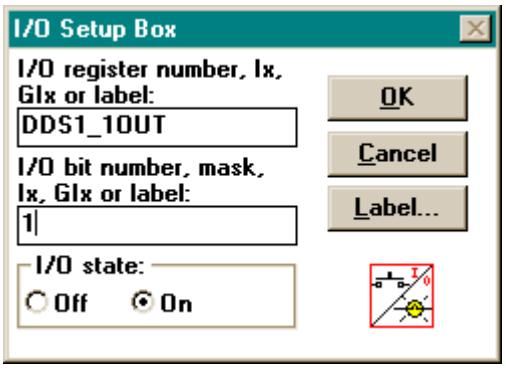
The **I/O Register** data entry box specifies the target register number. The entry can be a number, an integer variable, or a defined register label.

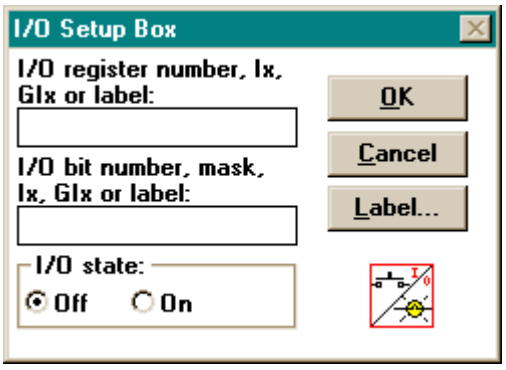
The **I/O Bit mask** data entry box specifies the bit or bits to be controlled and permits entry of an integer constant or equivalent label, as an "and" mask for the target register. A single bit is specified by a number, an integer variable, or a defined bit label. Multiple bits in a single register are specified in hexadecimal by a bit mask (e.g., 0x21 would specify bits 5 and 0).

Multiple bits can be changed with a single icon by entering an I/O bit mask that specifies more than one bit (e.g. 0x21 or a label equivalent to the desired mask). A pop-up dialog box of Bit Labels is available for adding, editing or deleting register bit labels.

The radio buttons for I/O state determine whether the target register bits, enabled by the I/O Bit mask, are cleared (logic zero, or off) or set (logic one, or on).

A pop-up dialog box of Register Labels is accessible, and permits adding, editing or deleting register labels. The Register Labels dialog box provides a scrolling list of default labels for the standard CLC system, axis, task, and DDS-2 drive I/O card control and status registers.



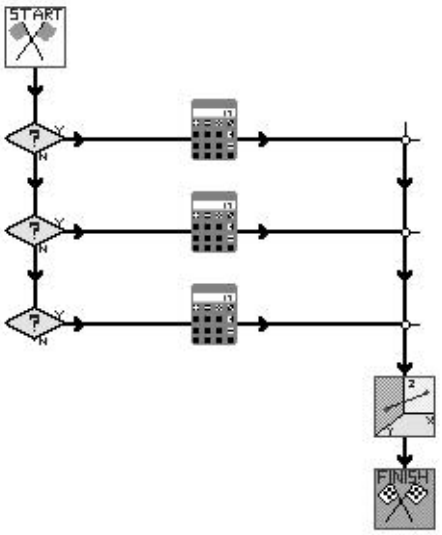


The example on the left would turn on output 1 of register 41 (DEA 4.2, Drive 1). The example on the right would turn on outputs 1 and 6 (bits 0 & 5) of register 41.

Join

A Join icon makes it possible to connect one line to another. VisualMotion icons have a maximum number of inputs; "join" overcomes this limitation by permitting many program flow paths to combine into a single path.

The Join icon is often the only method of completing a program flow, since VisualMotion cannot cross interconnecting lines. In addition, your program may require branching to several different calculations depending upon a certain condition. After the Calc icons you can use join icons to return to the main program flow before it enters the next icon.



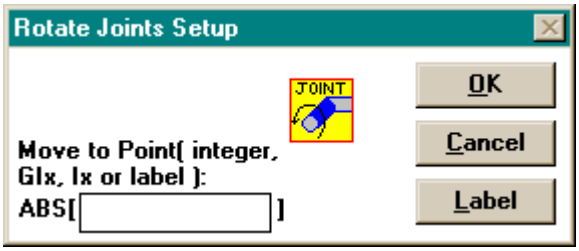
Joint

The Joint icon is used for point-to-point movement and joint (elbow) positioning, typical to robotic motion. It changes one or more motor angles from the current set J0 to a new set J1 defined by a absolute(ABS[x]) point. This instruction is only for coordinated motion whose kinematic supports joint angles. It can only be used with a six axis robot with a second frame of reference (more than just x, y, and z).

A Joint move is an absolute point-to-point move, with only the endpoint of the move specified. It is the most efficient type of move because the path calculated by the path planner is optimized to minimize time. A Joint move uses the axis' maximum accel and decel rates, while line and circle coordinated motion commands use Path Maximum percentages (defined in Task parameters) and Maximum Acceleration and Deceleration rates (defined in Axis parameters). Rate limiting is done based on which axis limits the most so no axis is violated.

The actual path taken to the specified point is not defined and may assume whatever form the path planner requires; however, once programmed the path is repeatable.

The destination is specified as an entry in the absolute point table as an integer constant, variable, global variable or an equivalent label.



Line

All icons in each task, subroutine and event function must be connected. The line icon is used to draw a line indicating program flow from one icon to another. Clicking on the beginning icon surrounds the icon with a box. Clicking on the ending icon automatically draws a line from the first to the second icon, with an arrowhead indicating the direction of program flow.

Under some circumstances VisualMotion may be unable to route a line and displays a "Connection could not be made, try connecting adjacent blocks" dialog box. Lines may be manually routed by clicking adjacent empty squares on the invisible workspace grid from the first icon to the second. A manually placed line may not cross another line; attempting to do so displays an error box.

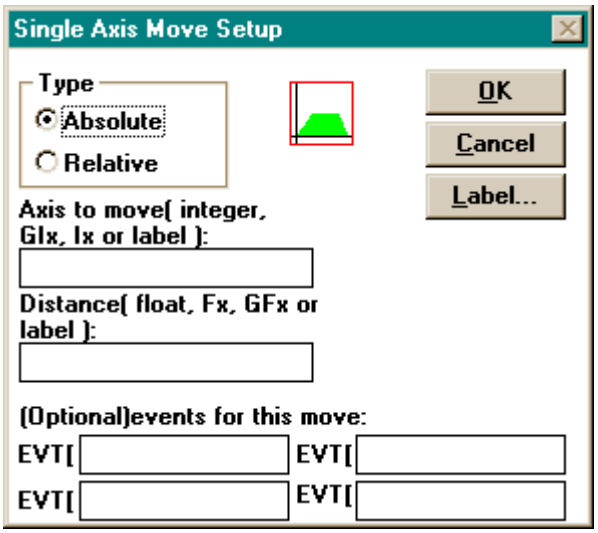
A line connecting two icons may be deleted by using the Scissor icon from the Utility palette. Simply select the Scissor icon, position the Scissor over the line to be deleted and press the left mouse button.

Move

The Move icon is used to program movement on any single non-coordinated axis from any task. The Move icon initiates motion only if the axis has been enabled previously with a “GO” icon. Placing a Move icon on a task or subroutine workspace automatically displays a Single Axis Move Setup dialog box.

The **Type** radio buttons specifies either an incremental move distance added to the current position (Relative) or a move to an absolute position (Absolute).

The **Axis to move** data entry box specifies the axis to move. It accepts an integer number, constant or variable.



The **Distance** data entry box specifies the incremental move distance or absolute target position. It accepts a floating point number, constant, or variable.

Up to four optional events (event functions) may be associated with the move. A data entry box for each of four possible events allows you to specify an event by a valid integer constant, variable (Ix), global variable (GIx), or an equivalent label, identifying the event. Any specified event functions must be programmed before running the program. The AxisEvt icon may also be used to enable events on an axis. The limit of four enabled events includes both Move and AxisEvt enabled events. The event types available for the “Move” icon are “Single Axis Distance from Start” and “Single Axis Distance from End”.

Axis Definition	**Units**
Single Axis Linear	inches
Single Axis Linear	mm
Single Axis Rotary	degrees (displayed as grads)
Coordinated	inches
Coordinated	mm
Ratioed (master)	see single axis definitions
Ratioed (slaves)	same as master

See Chapter 5. Programming Concepts - Events.

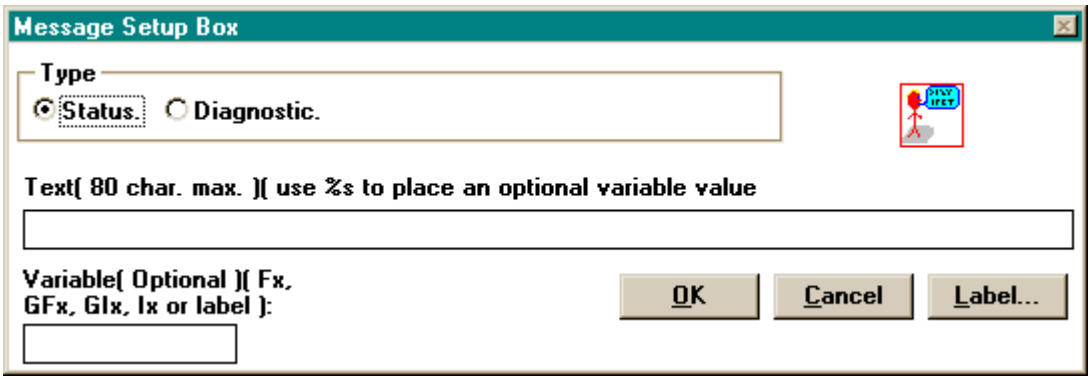
Msg

The Message icon is used to select a status or diagnostic message (up to eighty characters) at a specified point in the program flow. Placing a Msg icon on a task or subroutine workspace automatically displays a Message Setup dialog box.

Messages are used to inform a user about the current state of the program through Task Parameters T-0-0122 and T-0-0123, or through the teach pendant. Systems using Direct ASCII Communication may obtain messages through the RS-232 port. Messages may also be sent to the top line of the pendant and to the serial ports. Messages are available after the icon is executed in the program and remain in effect until another message of the same type is executed.

Status messages tell a user or machine operator something about the ongoing process. At run time, the current status and diagnostic messages can be viewed by selecting Tasks from VisualMotion's Status menu.

Diagnostic messages are typically used to provide information about the current state of the system, e.g., "412 No drives were found on ring." If an error occurs during task execution, this diagnostic message is overwritten with an error message.



An optional variable may also be displayed. This is useful for operator interface or debugging. Messages may have one formatted variable in its string by using “%s” as a place holder for the displayed variable.

Example: Message = "The current count is %s." *where %s equals variable I2 which has the label "current_count".*

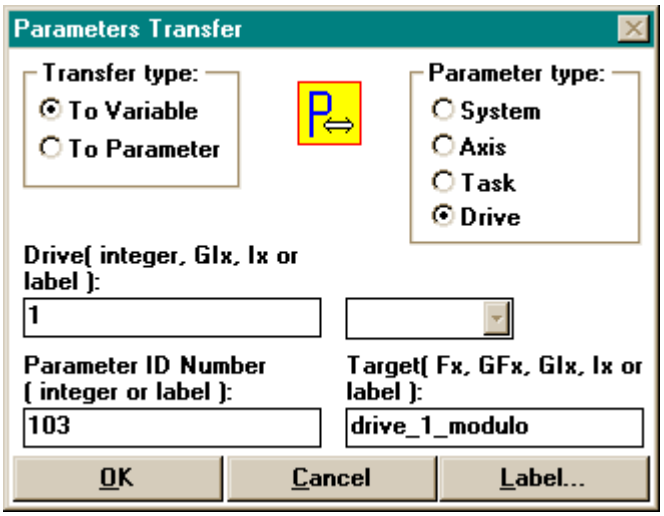
The displayed variable may also have a corresponding label. The label dialog box will be displayed when focus is on the variable field and cursor is doubled-clicked within the Message Setup Box.

It is also a good idea to copy the message into the comment box under the icon label. When the cursor is over the icon, the pop-up window will display the text message.

Param

The Param icon is used to transfer specified CLC or DDS drive-related parameters or variables between the PC Host system and an array of CLC variables. Transferring parameters to CLC-stored variables is the only way that programs can perform calculations and logical operations on parameter values.

Subsequently downloading the modified CLC variables permits dynamic modification of parameters during system operation. Placing a Param icon on a task or subroutine workspace automatically displays a Parameter Transfer dialog box.



Example 1: *To read the drive 1 modulo value (S-x-0103) into the variable "drive_1_modulo"*

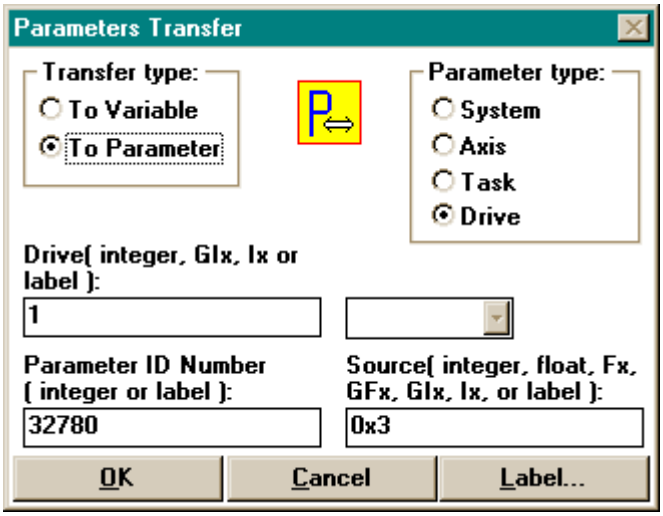
The radio buttons for Transfer type determine the direction of the transfer, from a specified CLC constant or variable to the parameter, or from the parameter to a CLC variable. The Parameter type radio buttons select from System, Axis, Task or Drive associated parameters.

Specify the parameter to be transferred in the Parameter ID Number data entry box as an integer or equivalent label.

If the **System** parameter type is selected, specify the source or target variable by entering an integer or floating point constant, variable (Ix or Fx), global variable (GIx or GFx), or an equivalent label in the appropriate data box.

If an **Axis** or **Drive** type parameter is selected, the additional data entry box enabled requires a valid integer constant, variable (Ix), global variable (GIx), or an equivalent label specifying the axis or drive ID.

If the **Task** parameter type is selected, a pull-down pick list (with scroll buttons) permits selection of one of the four CLC tasks (A, B, C or D).



Example 2: *To write to the drive 1 "Set Absolute Measuring Procedure Command" (P-x-0012)*

A pop-up User Defined Labels dialog box may be opened by clicking once within the appropriate data entry box, then double-clicking immediately outside the entry box.

Note: Certain system parameters are read-only and cannot be changed, while others may be modified only with the CLC system in "Parameter" mode. ***See the chapters on Parameters and CLC I/O (system status and control registers) for more information***. To write to and read from the product-specific drive parameters (P parameters), add 32768 to the P number.

Path

The Path icon is used to set up multi-axis coordinated straight line motion. Placing a Path icon on a task or subroutine workspace automatically displays a Coordinated Line Setup dialog box.

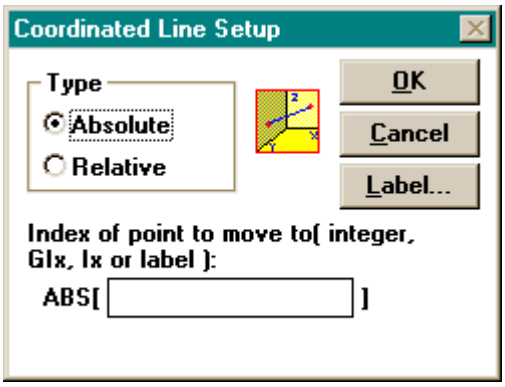
Motion may be Absolute or Relative and is defined by the two endpoints of the line of motion. Points are specified by an index into the point table using an integer constant, variable, global variable or an equivalent label. ***Refer to the Calc icon***. The second point, or target point, must be a point on the point table. The first point is the current position which can be anywhere.

An absolute move begins from the endpoint of the previous path segment, or current position if the system is halted, and terminates at the absolute point specified.

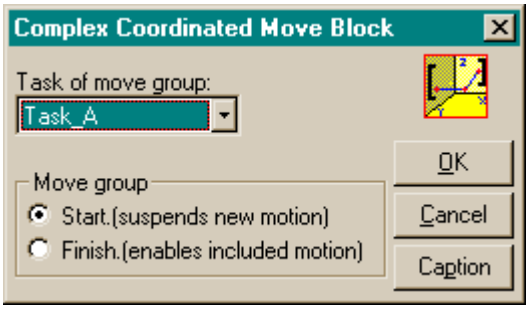
The relative move begins at the endpoint of the previous path segment, or current position if the system is halted, and terminates at the relative offset point specified.

A pop-up dialog box listing the User Define Labels is available for the ABS and REL data entry boxes.

A Go icon is not required when using the Path icon. When executed, the motion will immediately be sent to the path planner. Stepping to the next icon will take place immediately.



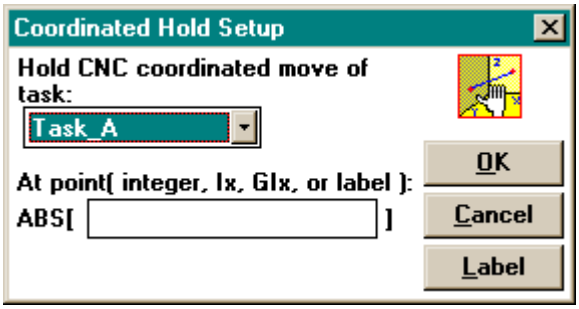
PathCalc



The PathCalc icon is used to set up a complex coordinated move block in the Trans 01D product. It is not supported in the GPS product.

Path/Hold

This icon allows motion to be stopped at a specific position within a CNC path planner. The position in ABS [k] is the path holding point within the selected Task. Approaching a stop point causes motion to enter controlled stop mode and decelerate until a resume is executed. If a resume is never executed, motion stops at ABS[k] with the stop state maintained. When resumed, acceleration to the programmed feed rate follows immediately.



If another, different Path/Hold is executed before the previous has completed, the new stop point is placed in a “first in/first out” queue and processed after the current one is completed.

Consecutive stop signs placed within a CNC step size are lumped together into one stop. Therefore consecutive stops placed closer than (SPD + ACC*DT)*DT units cannot be resolved by the motion generator. SPD is the feed rate for the segment, ACC is the acceleration for the segment and DT is 0.008 seconds.

User tasks are not blocked but continue executing normally so long as the stop queue is not full. If the queue is full, this instruction blocks until an element is removed from it.

This instruction does not effect normal hold/resume cycling nor does it effect pre-existing time and distance based events.

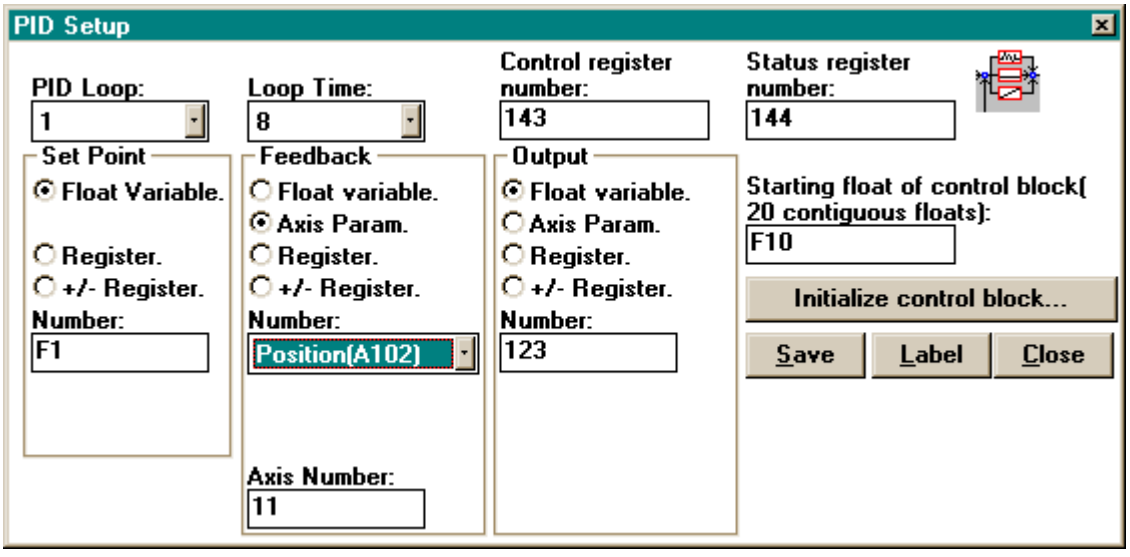
PID

The PID icon is used to initialize and install a proportional control loop on the CLC card, up to 10 loops are possible. The location of the PID icon in the program is not important, it is only executed at program activation. The program flow will be slowed if this icon is continuously processed, so it desirable to execute this icon only once per PID loop.

Set point can be a program or global variable(Fx, GFx, GIx, Ix, or label), or signed or unsigned register. Internally, the value is converted to a float. It's offset is then subtracted and the resultant multiplied by it's scalar.

Feedback and **output** may be a program or global variable(Fx, GFx, GIx, Ix, or label), a signed or unsigned register, or an axis parameter. Three axis parameters(position, velocity, and torque) are selectable in a list box. Other axis parameters can be added by selecting optional1 or optional2 and adding the axis parameter number.

The optional axis parameter data is added to the cyclic SERCOS data for update every SERCOS cycle time. If using the optional axis parameters, care must be taken to avoid using them for other purposes. Internally, the value is converted to float. It's offset is then subtracted and the resultant multiplied by it's scalar. A list of valid axis parameters can be viewed in drive parameter S-0-0188 "List of configurable Data in the MDT".

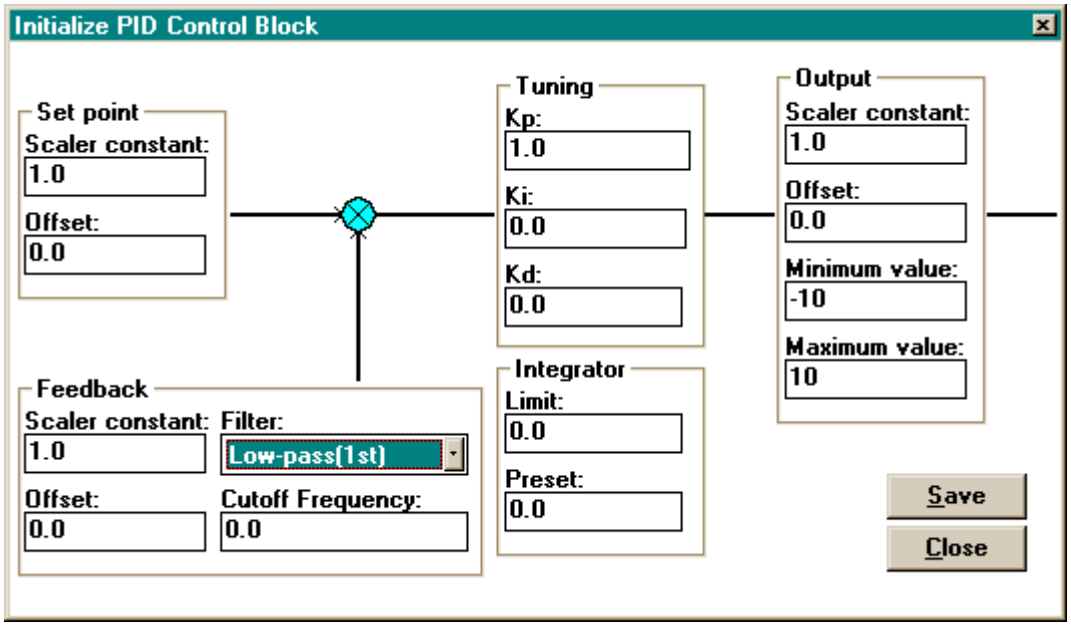


The Starting float of the control block is the first variable used to store the initialization values. The remaining variables are sequentially assigned to each control block value. (F11, F12. F13, F14.....)

The **control block** is a group of 20 floats used for:

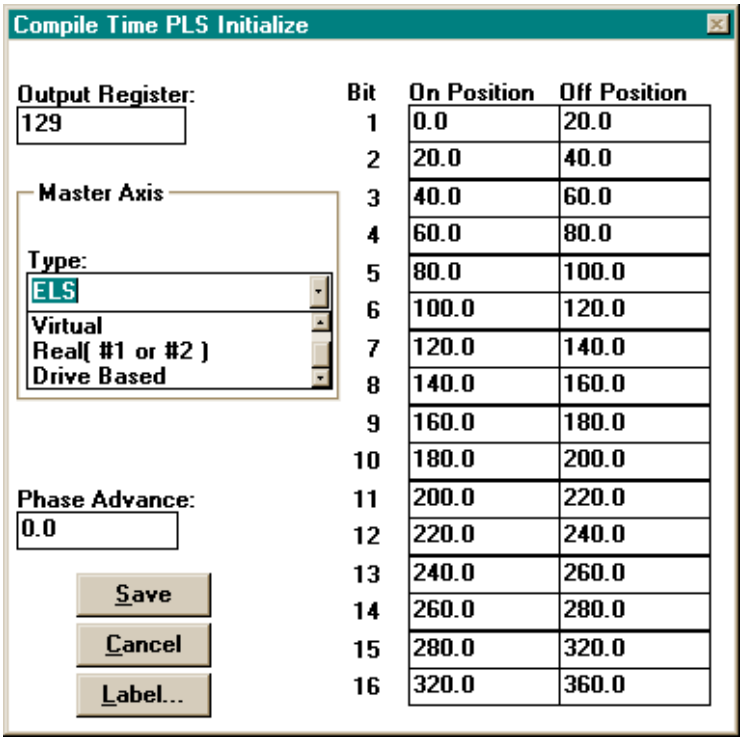
F(x)	Command scalar value	default 1.0.
F(x+1)	Command bias value	default 0.0.
F(x+2)	Feedback scalar value	default 1.0.
F(x+3)	Feedback bias value	default 0.0.
F(x+4)	Kp value	default 1.0.
F(x+5)	Ki value	default 0.0.
...	Kd value	default 0.0.
	Ki limit value	default 0.0.
	Minimum output value	default -10.0.
	Maximum output value	default 10.0.
	Preset value	default 0.0.
	Output scalar value	default 1.0.
F(x+12)	Output bias value	default 0.0.

Control block values, default or user selected, are added at compile time. If program floats are used, they must have been allocated in the Size Icon.

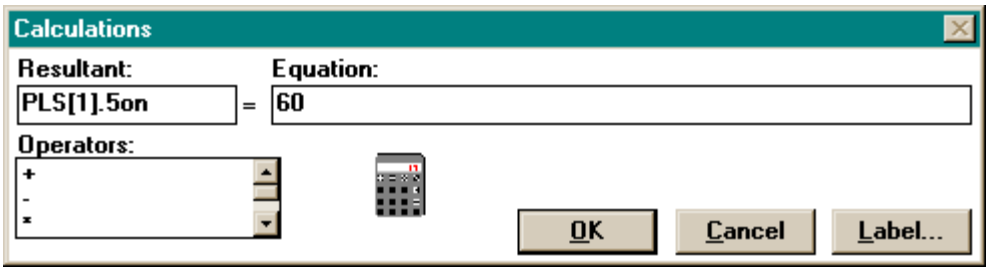


PLS

The PLS icon initializes the programmable limit switch data structure when the program is compiled. The location of the PLS icon in the program is not important since it is not executed when the program is run. If more than one PLS icon is in the program, the last one encountered, while compiling, will overwrite any previous ones. The PLS is disabled while the output register is zero or when all bits have zero in both the on and off positions. ***See Chapter 4. Visual Motion Menu Commands for more PLS information.***



A programmable limit switch supports 16 outputs and a phase advance. The phase advance is added to all positions in the PLS table. The position input for the PLS can be assigned to an ELS, Virtual, Real or Drive Based master. The outputs are updated every SERCOS cycle. It is possible to write to the positions in the PLS table in the user program with the ***Calc*** icon.

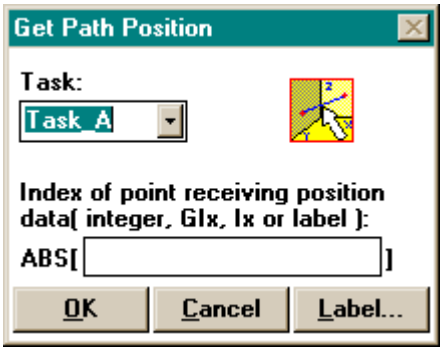


Position

The Position icon is used to obtain the current position of one of the tasks from the path planner. Once this position is read the value is stored in an absolute point table. This is useful when operating the system in a teach mode without a Teach Pendant. It also could be used to check axis position while running a program. For instance, a loop could be setup whereby the path position is constantly obtained and compared to a target position. When the path position matches the target position, a bit is toggled, causing a new branch test result which alters program flow.

Once this icon captures the current position into a point, ***Calc*** icons can be used to copy the attributes of these points to make a new ones. Single axis motion can be created from coordinated motion data and offset or speed change adjustments can be made automatically.

Placing a Position icon on a task or subroutine workspace automatically displays a Get Path Position dialog box. The Task pop-down menu permits selecting one of the four CLC tasks (A - D). The absolute point table destination for the position data may be specified by an integer constant, variable, global variable or an equivalent label. A pop-up User Defined Labels dialog box is available.



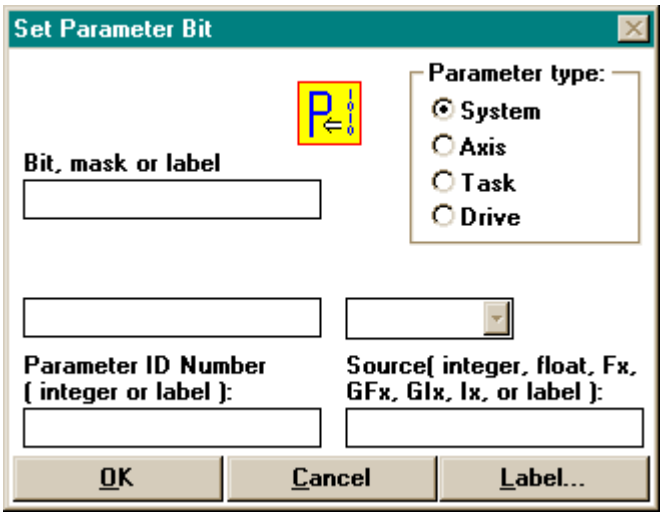
PrmBit

The Prmbit icon is used to set specific bits in system parameters that have a binary format. Parameters are specified by type(System, Axis, Task or Drive), group(axis number, drive number, or task) and number. System parameters have only type and number. For the product-specific drive parameters (P parameters), add 32768 to the P number. For specific ***parameter*** information see the CLC or Drive manual.

The source may be a float or integer constant, global or program (Fx, GFx, GIx, Ix) variable, or equivalent label.

Bits may be specified in decimal (i.e. 1, 5, 12) or hexadecimal form (i.e. 0x1000, 0x8097, 0x0001) or equivalent label.

NOTE: *The Prmbit icon cannot be used with parameters that are read-only or can only be written to in SERCOS phase 2.*



Prmint

The Prmint icon is used to initialize those CLC system, axis, drive, and task parameters that can only be written to in SERCOS phase 2. For the product-specific drive parameters (P parameters), add 32768 to the P number.

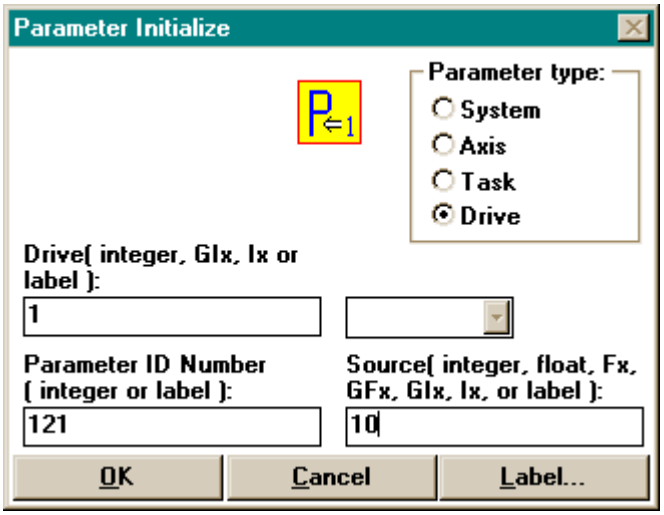
The parameter type is selected by choosing one of the Parameter type radio buttons.

Selecting Axis or Drive enables a data entry box, permitting entry of an integer constant, variable (Ix), global variable (GIx), or equivalent label specifying the axis or drive.

Selecting Task enables a pop-down menu with selections for the four CLC tasks.

The Parameter ID Number data entry box requires an integer constant or an equivalent label, specifying a valid number (in the range of 1 to 65535) for the parameter to be initialized.

The Source data entry box specifies the value to be loaded to the parameter. The type specified must match the type of the parameter, i.e. a floating point constant, variable (Fx), global variable (GFx), or equivalent label for a floating point value; an integer constant, variable (Ix), global variable (GIx), or an equivalent label.



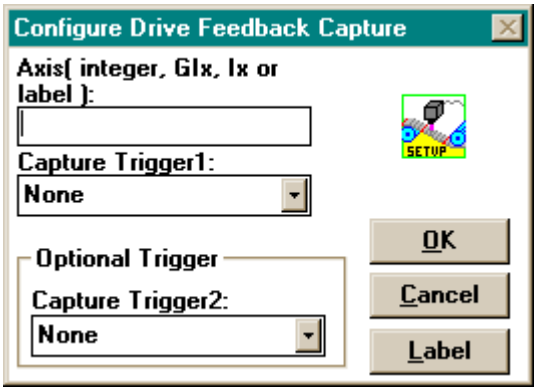
Example: *The gear ratio input revolutions of drive 1 is initialized to 10.*

NOTE: *When using variables as the parameter source, a SERCOS error may occur if no value has been entered for the variable as the system switches from phase 2 to phase 3.*

Probe

The Probe icon is used to enable drive based position capture inputs (Probe 1 and Probe2) for a drive (non-RAC) assigned to the specified axis. On the programmed transition, a flag is set in the cyclic SERCOS data informing the CLC of its happening. The ProbeEvt icon (***see Section*** ProbeEvt) assigns a event to execute when this flag is set.

This icon is necessary to put the captured probe position in the SERCOS telegram and it executes during the transition from Phase 2 to Phase 3. This icon should only be used once for each drive.



Axis - indicates the drive whose position will be captured on its Probe input transition. The entry may be an integer, global integer variable(GIx), program integer variable(Ix), or an equivalent label.

Trigger 1, Trigger 2 - enables drive based position capture for selected drive I/O input, Probe 1 or Probe 2. Capture can be on 0->1 or 1->0 transition.

One or two Capture Triggers may be enabled. Each may be assigned to none, or one of the two drive probe signals, with selection of trigger on rising or falling edge :

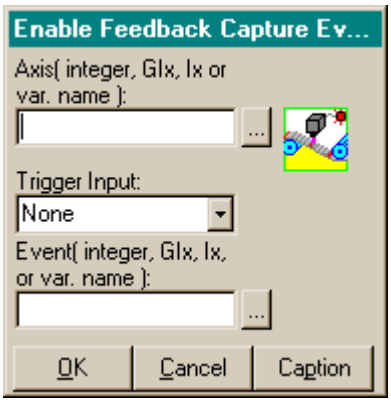
Data stored in this Parameter

Probe 1 low (0) to high (1)	S-x-0130
Probe 1 high (1) to low (0)	S-x-0131
Probe 2 low (0) to high (1)	S-x-0132
Probe 2 high (1) to low (0)	S-x-0133

Probe inputs to a DDS-2 drive are typically used as inputs for "in-position" switches or "feelers" associated with the drive axis.

ProbeEvt

The ProbeEvt icon is used to enable a CLC event for triggering by a DDS-2 drive probe transition. The event function must have been entered by choosing the Add Event Function from Visual Motion's Edit menu.



The Axis triggering the event is specified by an integer constant, variable (Ix), global variable (GIx), or an equivalent label. The corresponding probe trigger must be assigned to the same axis using an Axis or Probe icon.

The DDS-2 probe event used as the Event Trigger Input is selected from the pop-down menu. Choices include none (trigger disabled), Probe 1 low-to-high, Probe 1 high-to-low, Probe 2 low-to-high, and Probe 2 high-to-low.

The Event to be triggered on position capture occurrence is specified by an integer constant, variable (Ix), global variable (GIx), or an equivalent label that provides an index into the event table. The Event must have been allocated in the *Size* icon and initialized prior to occurrence.

The axis position at the time of the probe trigger is immediately available by reading the event's "a" field from the event table, (i.e., EVT[x].a), or by using the Param icon to read the corresponding probe's position capture from the drive as a drive parameter (Dx.00130 - Dx.00133). Because this instruction does not enable a repeating event, it must be executed for each new edge to be scanned.

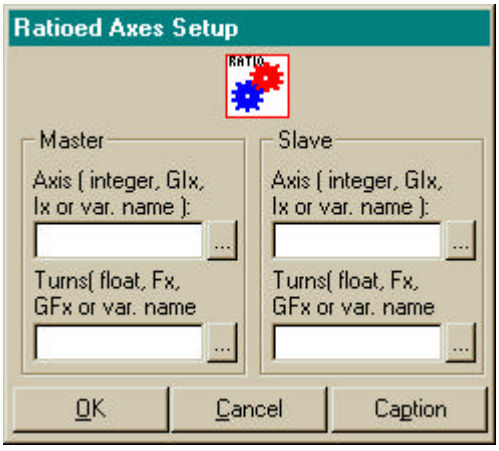
S-0-0130	Probe 1 0->1
S-0-0131	Probe 1 1->0
S-0-0132	Probe 2 0->1
S-0-0133	Probe 2 1->0

Ratio

The Ratio icon is used to set the ratio between two axes in a master/slave relationship, as when a gantry robot has a motor on each side of a supporting circular track. It can be used in Tasks which also contain coordinated axes.

The Ratio icon may also be used to link several axes to the same master axis or to chain several drives together. For example: if drive 1 is master to drive 2, drive 3 can be made a slave to drive 2, thereby linking drive 3 to drive 1 through drive 2. Note that the response time of drives chained in this manner is additive, at least one SERCOS cycle (approximately 2ms) must occur between each master to slave link.

Placing a Ratio icon on a task or subroutine workspace automatically displays a Ratioed Axis Setup dialog box**.** The Master and Slave Axis should already be assigned as ratioed axes in the Axis icon**.** ***See the Axis Icon for more information*****.**


Ratioed Axes Setup
Master
Axis (integer, GIx, Ix or var. name):
Turns(float, Fx, GFx or var. name
Slave
Axis (integer, GIx, Ix or var. name):
Turns(float, Fx, GFx or var. name
OK
Cancel
Caption

The **Master** and **Slave** axes are selected by entering an integer constant, variable, global variable or an equivalent label. The ELS Virtual Master, axis 0, may be used as the Master Axis. A pop-up User Defined Labels dialog box is available for all four data entry boxes. One axis is selected as a master axis. A slaved axis must not be assigned to any task other than the task containing the master axis.

Rotation of the master axis controls the proportional rotation of the selected slave axis according to the formula:

Slave axis velocity = Master axis velocity ´ (Slave ratio factor ¸ Master ratio factor)

The Master and Slave **Turns** data entry boxes permit simple entry of the ratio between the axes. The ratio factors may be a floating point constant, variable, global variable, or an equivalent label. Individual data boxes for

master and slave eliminate the need to normalize the ratio. For example, simply entering the number of teeth on each of two meshed gears allows VisualMotion to calculate the necessary coefficient. Each factor is in floating point format and is normalized before the division operation. This insures that the calculation maintains maximum precision with repeating decimals such as 2/3.

Entering the axis ratio factors using the Ratio icon automatically updates the master factor parameter A-0-0031 and slave factor parameter A-0-0032.

By default, the slave axis is maintained in the drive's position loop mode at all times, even when the user program is not running. The drive's shaft position remains locked to the master axis within the torque limits of the drive.

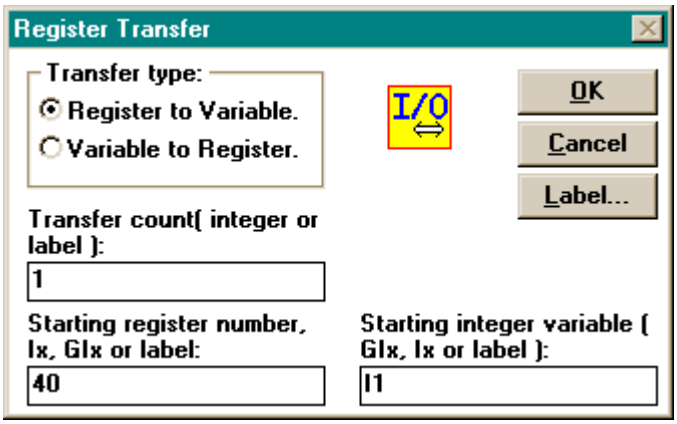
Follow ELS Master Axis

Optionally (through parameter A-0-0038), the slave axis may follow the velocity feedback of the master or be maintained in velocity mode instead of position mode. An ELS master may be used as the master axis, by choosing axis 0.

A ratio mode slave can now follow the ELS master (virtual or real). On a system without ELS drives, all slaves can follow the ELS master so that there is no lag between slaves. To follow the ELS master, enter 0 for the master axis in the axis_setup icon or axis/setup text language command. If no ELS slave axes are present, it is necessary to set the following ELS parameters using parameter/init statements: C-0-0150, C-0-0151, C-0-0152, C-0-1000, C-0-1001.

Register Transfer

The Reg icon is used to transfer data between the I/O registers and either the integer variable or global integer variable table. One to 1024 registers (16-bits each) may be transferred with a single command. Placing a Reg icon on a task or subroutine workspace automatically displays a Register Transfer dialog box.



Example: *This transfers a 15 bit DEA card input register to a variable.*

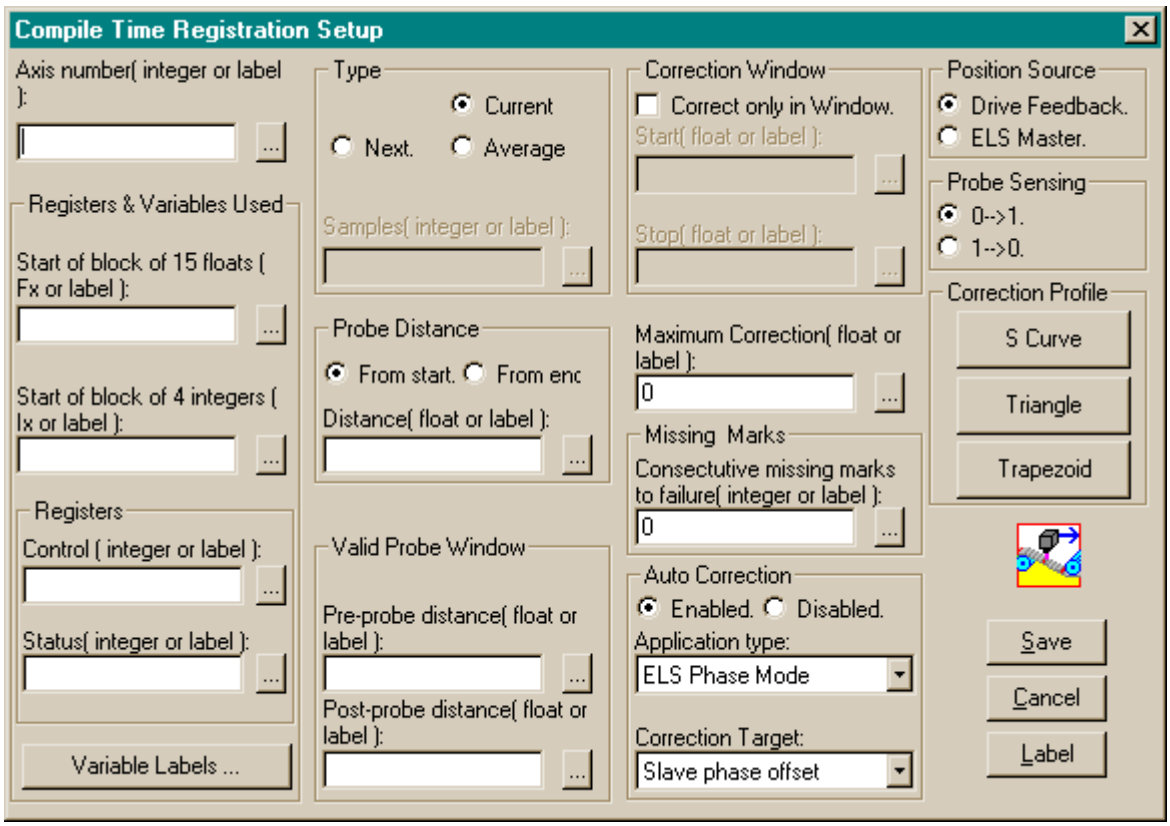
Transfer type selects the direction of data transfer between the I/O register(s) and a CLC integer variable table. Transfer count specifies the number of consecutive 16-bit words to be transferred. **Starting register number** specifies the base, or lowest address of the source for the start of the transfer. **Starting integer variable** specifies the base address of the target of the transfer.

The **Transfer count**, Starting register number and starting integer variable must be an integer constant, variable (Ix), global variable (GIx), or an equivalent label.

Registration (Not Functional with GPS 6.0)

This icon is used to initialize an auto registration procedure. Registration is the process of referencing a product edge or register mark. This allows positioning errors to be detected and corrected before an upstream (web, product) or downstream (die-cutter, print cylinder, etc.) process takes place, depending on the machine design.

This function tightly couples the CLC system with the **Probe Event** on the drive that the product is being referenced to. Registration is accomplished by comparing the captured position to a target value and correcting for the difference. The registration error is automatically assimilated into the axis motion profile, providing a smooth, seamless correction. The registration error can be corrected using S-curve, Triangular, and Trapezoidal correction profiles.



Functional Description

The Registration function integrates several tasks:

- It automatically executes a high priority process that is triggered by an axis probe event.
- This process generates a *registration error* based on the settings of the drive, system and registration parameters.
- The CLC will then automatically reconcile details such as rollover, probe On/Off position window, setting drive parameters (edge, arming, etc…), and routing the correction data to the appropriate axes.

The registration function is compatible with axes defined in ELS Phase, Drive Cam, and CLC Cam (table or indexer) modes. The values needed for registration are read from and written to the CLC with user program variables. Up to 4 axes can use the registration function.

The Registration icon dialog box assigns variables and I-O registers, and initializes axis parameters. All values are assigned to blocks of floating point and integer variables. This allows them to be read or written from a user interface or a PLC with no additional program code.

When the Cam Indexer is used, it performs the registration correction with a profile based on the *Correction Distance*. For other types of axes, a relative phase offset (either master or slave) is sent to the drive, and the drive performs the correction. The phase offset, equal to *Correction Distance*, is applied according to the *Correction Procedure* and *Correction Maximum* parameters.

The registration function can be used with the following Indramat drive firmware versions:

DIAX03/04 - ELS-04VRS

DIAX03/04 - ELS-05VRS

ECODRIVE- SSE-03VRS

Registration Parameters

The following Parameters are used by the registration function. Most of them are written by the user with the registration icon dialog box. Some of them are also calculated and written by the CLC card or the Drive.

Parameters	Description	Variable	Written by
Type	Current, Next, Average	Int 0 bits 0-1	user
Probe Distance from Start/End	Select setpoint to be calculated from start/end of move	Int 0 bit 2	user
Probe Distance	Expected position of mark from start or end of move	Float 0	user
• Mark Setpoint	Absolute expected position of mark to calculate error	Float 6	CLC
• Correction Distance	Correction distance (Error amount)	Float 7	CLC
• Probe Value	Feedback position at probe 1 trigger	Float 8	drive
Valid Probe Window Start	Distance before Mark Setpoint to look for mark	Float 1	user
Valid Probe Window End	Distance after Mark Setpoint to stop looking	Float 2	user
Correction Window	Correct only in the correction window.	Int 0 bit 12	user
Correction Window Start	Correction to start at this point	Float 3	user
Correction Window Stop	Correction to be completed before this point	Float 4	user
Maximum Correction	Maximum correction per period	Float 5	user
Missing Marks	Number of missed marks allowed	Int 1	user
Auto Correction	Enable correction sent to phase adjust, etc.	Int 0 bit 8	user
Correction Target	Select correction target: master offset, slave offset, etc.	Int 0 bit 9-10	user
Position Source	Measure Master or Slave position	Int 0 bit 4	user
Probe Sensing	Use Leading/Trailing edge of mark	Int 0 bit 3	user
Correction Profile	Shape of the correction cam	Int 2	user

Type

This selects the desired type of registration correction calculation to be used.

- Current:
- Next:
- Average:

Samples (Integer or Label) -

Probe Distance from Start/End

This bit determines if the *Mark Distance* is relative to the Start or End of move. This option applies to cam axes that use the indexer feature. For other modes, the mark distance is an absolute position.

0 - End of Move

1 - Start of Move

Probe Distance

This value is used to determine the calculated position of the axes that a mark is expected to detected.

Cam Indexer:

Based on the setting of *Mark at S/E* bit, the *Mark Setpoint* value is automatically calculated as follows. *Starting Position* is the position of the axis at the start of the index. *Ending Position* is the expected position at the end of the index (minus registration correction).

If Start of move: *Mark Setpoint* **=** *Starting Position* **+** *Mark Distance*

If End of move: *Mark Setpoint* **=** *Ending Position* **-** *Mark Distance*

ELS Phase, CLC Table Cam, or Drive Cam Mode:

The *Probe Setpoint* is an absolute position on either the master or the slave (set via an integer option bit).

Probe Setpoint = Probe Distance

Mark Setpoint: This value is used to calculate registration error and various position points used in the registration process. See Correction Distance, Probe Window On/Off.

Correction Distance: This is the amount of correction distance that will be applied to the current index in units. Based on the selected *Correction Profile* type this error is added to the cam profile or ELS slave position at the position defined by the correction window. Only one value is read during each index cycle.

Correction Distance = Mark Setpoint - Probe Value

In future versions, correction distance may be optionally buffered through several cycles in case the measurement is upstream, or may be averaged or filtered for a smoother correction.

Probe Value: This value indicates the measured probe position value that was captured by the drive when the registration mark was detected on the probe 1 input. It is saved only when the axis is within the probe window. It is used in the calculations associated with determining registration error.

Valid Probe Window Start/End

This window selects the period within the move cycle that the registration mark can be read. The probe position is captured and the correction distance is calculated only when the mark is found within this position range. The probe window is related to the selected measurement value (master or axis position).

Correction Window

This bit enables the period within the move cycle that the correction can be applied(defined by “Correction Window Start/End”). If this bit is clear, the correction is applied immediately after the probe position is captured.

Correction Window Start/Stop

This window selects the period in the move cycle to apply the registration correction. If the option *Correction Window Enabled* is not selected, the correction is applied immediately after the probe position value is captured. Otherwise, the correction starts at *Correction Window Start*, and must be finished before *Correction Window End*.

When using the cam indexer, the start and end position in relation to the master axis are entered in these parameters. Correction motion will only take place during this window. If the probe comes in during the correction window, say at position x of the master, then the maximum allowable correction during that cycle is calculated as follows:

(maximum correction)(end - x)/(end - start)

For other motion types, the positions are related to the measured value (master or axis position). The correction is added to the value selected in the *Correction Target* option. The drive handles the phase offset internally, based on its velocity, acceleration, and/or filter parameters. It is necessary to set these drive parameters so that the correction move finishes before the next cycle.

Maximum Correction

This is the maximum amount of correction distance that will be applied to the current index in units. If zero, the full amount of the correction is used for the current index. If a value is entered, the *Correction Distance* value is compared to *Correction Maximum* and correction up to this value will be applied. The remainder will be discarded.

Missing Marks

This defines the number of consecutive missed marks that are allowed before the CLC sets the Max Missed Marks status bit.

Auto Correction

By default, the CLC sends the *Correction Distance* to the drive to perform a phase offset, based on the correction window and correction target. It is possible to disable automatic correction, so that it can be handled in the user program. This option applies to all motion types except for the cam indexer, for which automatic correction is always enabled.

0	-	Enable Automatic Correction
1	-	Disable Automatic Correction

Correction Target

These bits select the target parameter for the *Correction Distance.* When using the cam indexer, the correction uses a cam which is added to the cam position generated by the indexer. For other modes, the following selections determine the affected parameter:

Bit 10	Bit 9	ELS Phase Sync	ELS Velocity Sync	Drive Cam	CLC Cam (Table)
0	0	slave phase offset (S-0-0048)	additive velocity command (S-0-0037)	master phase offset (P-0-0061)	master phase offset (A-0-157)
0	1	invalid	ratio fine adjust (P-0-0083)	slave phase offset (S-0-0048)	slave phase offset (A-0-162)
1	0	invalid	invalid	cam shaft distance (S-0-0093)	Cam H factor (A-0-33)

Note: Automatic registration on ELS velocity synchronization axes is not yet supported in GPS-06.

Position Source

This bit determines if the slave axis (0) or ELS master (1) position is used in the registration correction calculation. This option applies to ELS, CLC cam (no indexer), and drive cam modes only. With the cam indexer, only the slave position can be measured. If slave position is enabled, the *Probe Value* captured by the drive is the axis feedback position. If master position is enabled, the *Probe Value* is the ELS master position.

0 - Axis Feedback Position

1 - ELS Master Position

Probe Sensing

This bit determines if the drive captures the axis position at the leading or trailing edge of the mark. The leading edge is indicated by a high (0-1 transition) on the probe input.

0 - Leading Edge

1 - Trailing Edge

Correction Profile

When using the cam indexer, this value defines the shape of the correction cam. Current selections are: S-curve, Triangular, and Trapezoidal.

Registration Control and Status Registers

The control and status registers that are used for each registration axis are allocated in the registration user program command.

Registration Control Register

9 - Enable Preset

10 - Enable Registration

Registration Status Register

9 - Correction at Max

10 - reserved

11- Mark in Window

12- Preset Enabled

13 - Registration Enabled

14 - Max Missed Marks

Enable Preset

The function of this bit depends on the selected correction procedure. If buffering or averaging is used (future versions), it resets the FIFO buffer and/or the average. Otherwise, this bit is ignored by the CLC.

Enable Registration

Set this bit to (1) to enable the registration function. All relevant parameters will be initialized, then the 'Registration Enabled' bit in the registration status register will be set. To disable the registration function, set this bit to (0).

Correction at Max

This bit is set to (1) when the *Correction Distance* exceeds the *Correction Max* parameter. It is reset to 0 after the next mark is found and *Correction Distance* is less than or equal to *Correction Max.*

Mark in Window

This bit is set to (1) as soon as a registration mark has been detected within the probe window. It is cleared when registration is first enabled, or on the following cycle when no registration mark is found in the window. This bit can be mapped to a task input or system input event, so that the user can run additional program code when the mark is detected.

Preset Enabled

This bit is an acknowledgment of the Enable Preset control bit.

Registration Enabled

This bit is an acknowledgment of the Enable Registration control bit. Its value will not match that of the control bit until the CLC has completed the process of enabling or disabling registration. This process could take several milliseconds, depending on the parameters that need to be initialized.

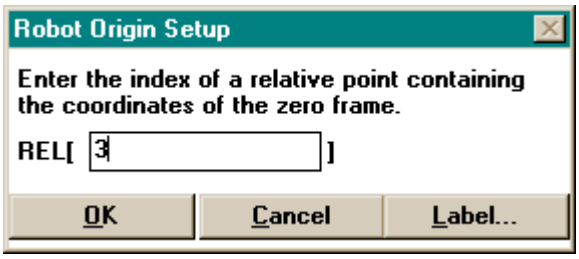
Max Missed Marks

A missed mark counter is incremented when the probe window is traversed without a mark being found. This bit is set when the counter exceeds the *Missed Marks* parameter. The user program or PLC can then take appropriate action or alert the user of this condition. This bit is reset by the CLC only upon a 1-0 transition of the Registration Enable bit.

RobotOrg

The RobotOrg icon is used in coordinated motion programs to construct a zero frame of reference from the x, y, z, roll, pitch and yaw coordinates of a relative point. This moves the effective origin of the robot from the default to the location specified by the programmed relative point.

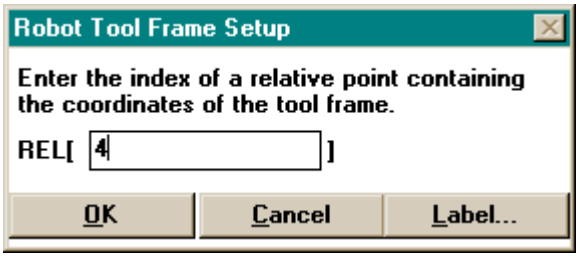
For example, if REL[3] = {1, 2, 3, 0, 0, 0, ...} and this point is specified as the robot origin, then the robot origin would be offset by one unit along the x axis, two units along the y axis and three units along the z axis. Once the icon instruction is executed, all jogging, teaching and path locations are affected by the new origin.



RobotTool

The RobotTool icon is used in coordinated motion programs to construct a tool frame of reference from the x, y, z, roll, pitch and yaw coordinates of a relative point. This moves the effective end-of-arm tool to the location specified by the programmed relative point.

For example, if REL[4] = {0, 0, 10, 0, 0, 0, ...} and this point is specified as the robot tool location, then the robot tool location would be offset by ten units along the z axis from the faceplate of the robot. Once the icon instruction is executed, all jogging, teaching and path locations are affected by the end-of-arm tool location.


Robot Tool Frame Setup
Enter the index of a relative point containing the coordinates of the tool frame.
REL[4]
OK
Cancel
Label...

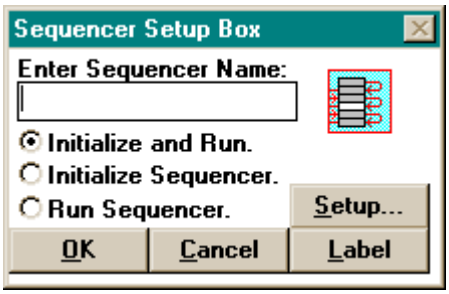
Scissor

The Scissor icon is used to delete a line between two icons. Select the Scissor on the palette, position the tip of the Scissor cursor over the line to delete, press the left mouse button to delete.

Sequencer

The Sequencer icon is used to initialize and/or run a Sequencer list. A Sequencer list contains user-defined steps that can be edited "on-line" without having to edit, compile and download a program. Each step contains a list of functions that will be executed sequentially following the order of the list. Each step also can have up to five function argument which will be passed to the subroutine when the function is executed. ***Refer to Chapter 5. Programming Concepts*** for more information.

The Sequencer name is a number or label equating to a number. The number has a range of 1 to n, where n is the number of sequences defined in the Size icon. If the Sequencer is only to run, it's number can be specified by a program or global integer(Ix, GIx).

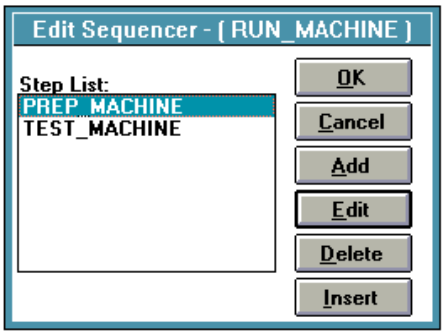


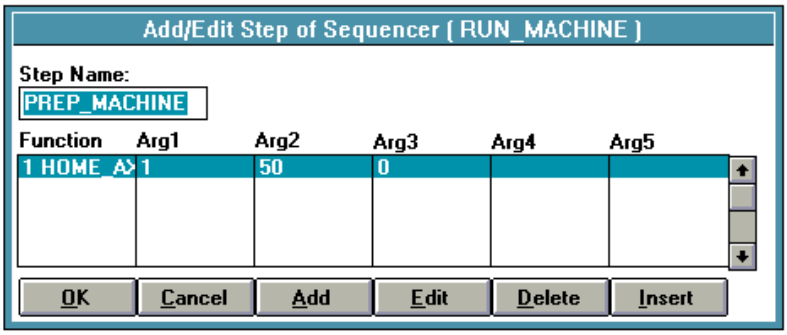
If Run Sequencer is selected the Setup... option is inactive and the icon can run a Sequencer that has previously been setup using the Sequencer Editor. Select "Run Sequencer" to just run the Sequencer without initializing it back to a default.

If *Initialize Sequencer* or *Initialize and Run* is selected, the Sequencer can be Setup... by the Sequencer icon according to the following icon dialog boxes. This is a good tool to use for restoring your program to a working default. When this icon is executed the setup information will overwrite any previous "on-line" changes that may have been made using the Sequencer Editor (***See Chapter 4. Visual Motion Menu Commands***). The system will stay on the sequencer icon until the sequence is complete.

When Setup... is selected the following dialog boxes can be used to add, edit, delete or insert step lists within a Sequencer.

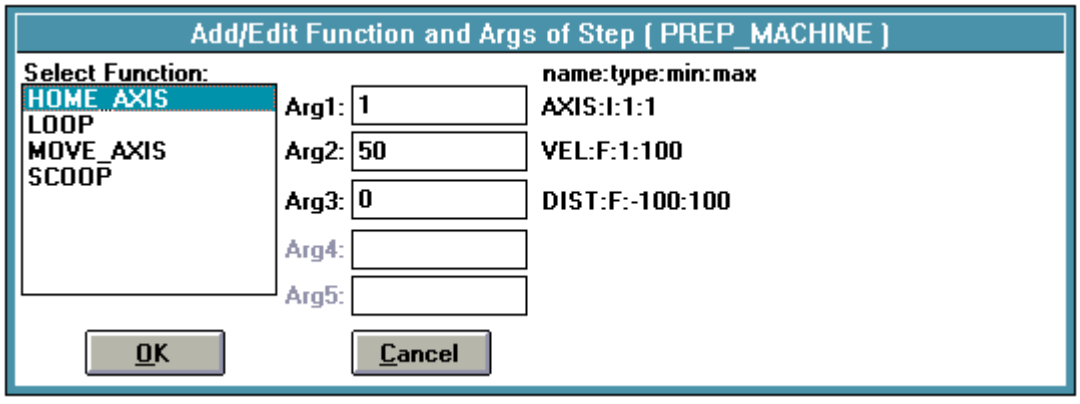
After selecting Add, a Step Name can be entered and edited as illustrated below. Each step has room for a twenty character name to uniquely identify it.





Functions need to have the "Callable from Sequence" access flag set in the ***Start*** icon of the subroutine before they can be added to a Step List. Function arguments must also be defined in the Start icon for them to be used.

After selecting Add, a function can be entered and edited as illustrated below. The function argument values must be within the limits predefined in the START icon of the subroutine. See ***Sub*** icon and ***Chapter 5. Programming Concepts***.



Size

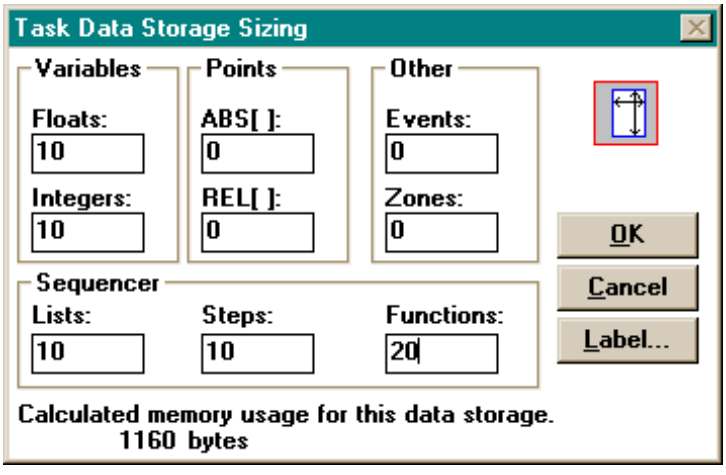
The Size icon allocates CLC memory for the different types of data storage each task requires. Placing a Size icon in the workspace opens a Task Data Storage Sizing dialog box. The entered values are the maximum number of variables, points, events, zones and Sequencer components that can be used in all the tasks.

The Sequencer components consist of lists, steps and functions. The values entered for lists and steps represent the total number of *different* lists or steps that are allowed in a program. If the same step is used more than once, in one or more lists, it still only counts as one towards the total. The total number of functions represent every time a function is executed in a step, even if the same function is used more than once. Each function will add to the total every time it is used. ***See Chapter 5. Programming Concepts for more information.***

The Size icon can only be used in the four main tasks; it cannot be used within a subroutine or event. In most cases, only one Size icon needs to be used per program. A size icon should be used in each Task that contains coordinated motion. When used, the Size icon should be placed immediately after the Start icon.

A program has a default value of fifty entries for integer and float variables. If any task uses more than 50 entries of any type, the Size icon must be used to allocate additional space. If a program uses less than fifty entries of any type, using the Size icon to reduce the allocation from the default to the actual number of data entries saves CLC memory space for other programs.

Variables each require 4 bytes, points 42 bytes, and events 118 bytes. The byte total of allocated memory is shown at the bottom of Tasks Data Storage Sizing dialog box. After a new value is entered and the cursor is moved outside the dialog box, a new byte total is calculated and displayed.

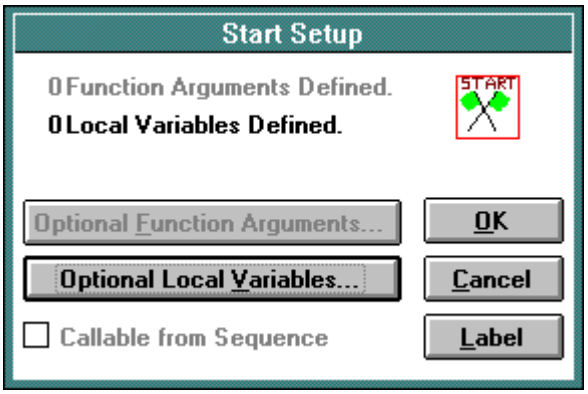


Start

The four CLC tasks (A through D), subroutines and event functions must all contain a Start icon. The Start icon indicates the beginning of program flow to the CLC compiler and also declares function arguments and local variables. The Start icon is equivalent to the CLC Text Language TASK/START command statement.

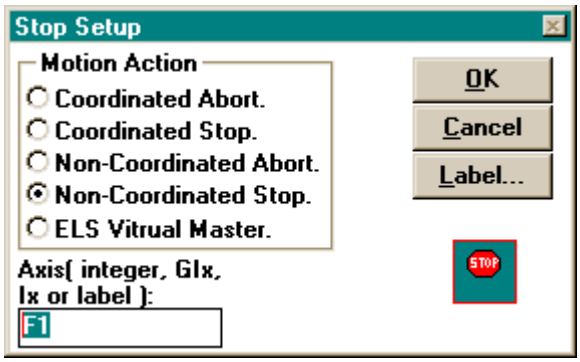
Stack based variables(function arguments and local variables) exist only while in the function(task, subroutines, or event) where they are declared. *Function arguments are those called from a subroutine icon or sequencer, and local variables are used within the subroutine for local data only they don't exist outside the subroutine.* This type of variable is useful for temporary results within a function or to pass values to a function.

The same label may be used for a function argument or local variable in different functions. The label may not be a keyword or other label(user, register, or bit). The total number of function arguments and local variables in a function is limited to 16. Up to 5 function arguments may be passed to a subroutine.



Stop

The Stop icon is used to halt motion on one or all axes used in a task. It will return the drives to an AH (Drive halt) state. Placing a Stop icon on a task or subroutine workspace automatically displays a Stop Setup dialog box.



A pop-up “User Defined Labels” dialog box is available by highlighting variables (F1) and double clicking inside the “Stop Setup” dialog box.

Coordinated Abort and **Coordinated Stop** decelerate motion on path, stopping all axes associated with the coordinated motion. Selecting Coordinated Abort or Coordinated Stop enables a pop-down menu permitting selection of one of the four CLC tasks.

Restarting motion after a Coordinated Abort requires toggling the Cycle/Start bit of the associated Task Control register. Motion stopped using a Coordinated Stop may be resumed with a Go icon, although resuming timed events that are programmed for motion at operating speed may result in events occurring at unexpected times.

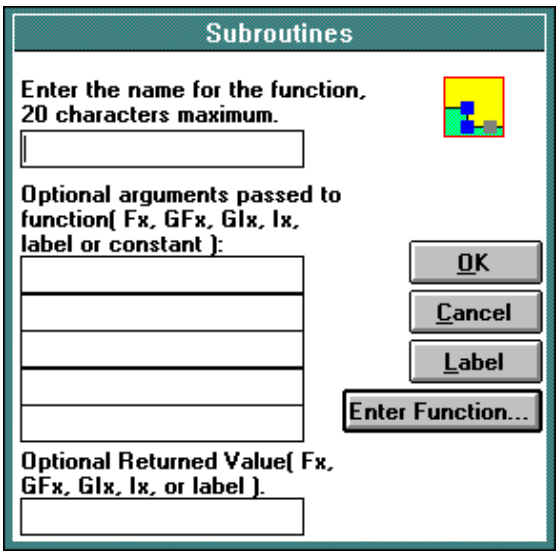
Non-Coordinated abort and Non-Coordinated Stop stops motion on the specified axis by decelerating the axis to zero velocity using the currently programmed rate. The axis to stop may be specified by a valid integer constant, variable, global variable or an equivalent label. Specifying a "-1" stops all axes in the task.

NOTE: *After an **Abort**, all queued events and the "look-ahead" motion calculated by the path planner are lost and the current move is aborted. The target position is set equal to the current feedback position. After a **Stop**, both queued events and the calculated path are retained. The target position is not set equal to the feedback position. When the axis is again enabled, it will complete the last commanded move.*

ELS Virtual Master decelerates the virtual master and all assigned axes to a stop at a point specified by an index into the point table using an integer constant, variable, global variable or an equivalent label. Multiple revolutions will be traversed if necessary to maintain programmed deceleration . To stop the virtual master without specifying a position, choose **Motion Action Non-Coordinated** and enter a “0” for the axis number.

Sub

The Sub icon is used to enter a subroutine function from the normal program flow. Placing a Sub icon on a task or subroutine workspace automatically displays a Subroutine dialog box.


Subroutines
Enter the name for the function, 20 characters maximum.
Optional arguments passed to function(Fx, GFx, GIx, Ix, label or constant):
OK
Cancel
Label
Enter Function...
Optional Returned Value(Fx, GFx, GIx, Ix, or label).

After you enter a name for your subroutine, clicking on the Enter Function button automatically replaces your current task or subroutine workspace with a new subroutine workspace in the main Visual Motion window. The title bar of the main window changes to show the name of the new subroutine. You can now use Visual Motion icons to build a subroutine in exactly the same manner as building a main Task. To return from a subroutine workspace to one of the four tasks, select the task from the View menu.

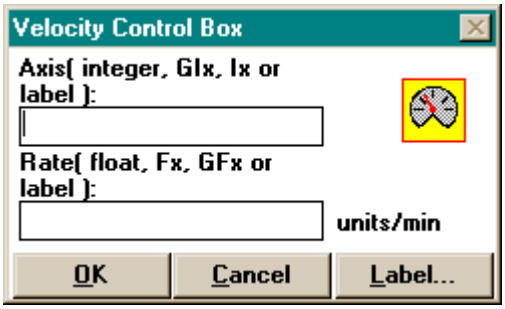
Function arguments are stack based variables passed to a subroutine. The setup dialog box for the Sub icon has entries for the function arguments. Up to 5 arguments may be passed to a subroutine. The arguments can be constants or variables (float, integer, ABS index or REL index). The function arguments need to be declared in the subroutine Start icon dialog box. A minimum and maximum value must also be entered for each of these arguments. This limits the range of argument values used during Sequencer editing from the teach pendant or the Sequencer interface.

Function arguments are not allowed in tasks or events, however, an optional return argument may be passed back to a task from a subroutine. A return argument is a good way of getting a return value from a subroutine. A common subroutine can then be utilized from more than one task. Multiple variables have similar states depending on conditions. You can call a subroutine with a variable being the return value to set its state.

Local variables are stack based variables used in a function. The total number of local variables and function arguments is limited to 16. Local variables, floats and integers are initialized to zero on each entry into the function. This type of variable can be used to store temporary results in a re-entrant function called on by more than one task. Local variables can also be used to avoid possible conflicts with program or global variables between tasks.

Velocity

The Velocity icon is used to specify a rate for motion on a single non-coordinated axis or the ELS Virtual Master (Axis 0). The axis may be specified by a valid integer constant, variable, global variable or an equivalent label. The velocity may be entered as a floating point constant, variable, global variable or an equivalent label. A pop-up dialog box of User Defined Labels is available for both data entry boxes.



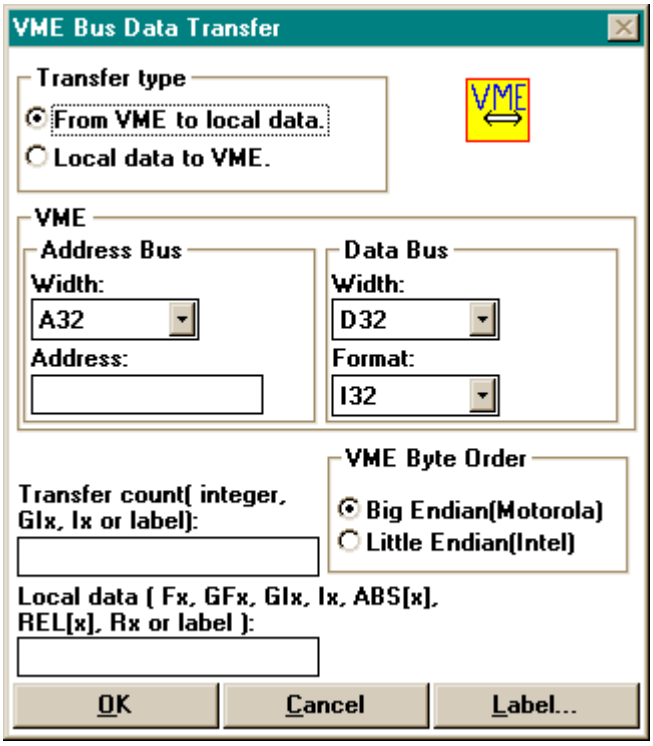
Units:

Single Axis Linear inches:	inches/min
Single Axis Linear mm:	mm/min
Single Axis Rotary:	RPM
Velocity Mode:	RPM
Virtual Master:	RPM

VME

The VME icon is used to transfer CLC data, such as variable and point tables, between the CLC's local memory and the VME address space.

Before using a VME Bus Data Transfer, the CLC interface with the VME bus must be configured for your VME system using the VME Configure selection from Visual Motion's Setup menu.



The Transfer type radio buttons select the direction of the data transfer operation. Data may be transferred between the CLC's local memory (point tables or variables) and the VME global memory address space that is accessible to any VME card in the VME card rack.

The VME section of the dialog box allows setting the base address, the address width, data width, and data format of the data within the VME system memory space. These settings must match your VME space or runtime errors will occur. In some cases when the data format of the source and target don't match, source data is sign extended or chopped to fit into the target.

Address Bus Width provides a pop-down menu allowing selection of A32/A24 (both 32 bit, 4 gigabyte and 24 bit, 16 megabyte), or A16 (16 bit, 64 kilobyte) VME memory address space.

Address specifies the starting address for the transfer operation.

Data Bus Width provides a pop-down menu allowing selection of D8 (byte), D16 (word), or D32 (double word) data width for the transfer with VME memory.

Format permits setting the data format of the VME source or target for the transfer. Available formats include 32 bit floating point; 8, 16, or 32 bit integer; 8, 16, or 32 bit unsigned integer; or point for transferring the multibyte data contained in an absolute or relative point table entry. Data is automatically converted to the data type of the specified target.

Transfer count specifies the number of data items to transfer with a single VME icon.

Byte Order sets the high-byte, low-byte order of multi-byte data (words, double-words, etc.). The byte order must be set to be compatible with the target processor on the VME bus that will be accessing the data. I.E., Motorola (the CLC uses the Motorola 68000 series) or Intel (such as the 80x86 and Pentium).

Local data specifies the starting, or base address of the CLC data to be transferred. Local data may be specified as an integer index to the CLC's integer or floating point variables (Ix or Fx), or global variables (GIx or GFx), or an integer index to the CLC's absolute or relative point tables. You may also use a valid equivalent label (a pop-up User Defined Labels dialog box is available). You may also send a constant to VME memory by specifying an integer or floating point constant, or equivalent label as the Local data.

The Data Bus Format above must be set to match the format of the Local data.

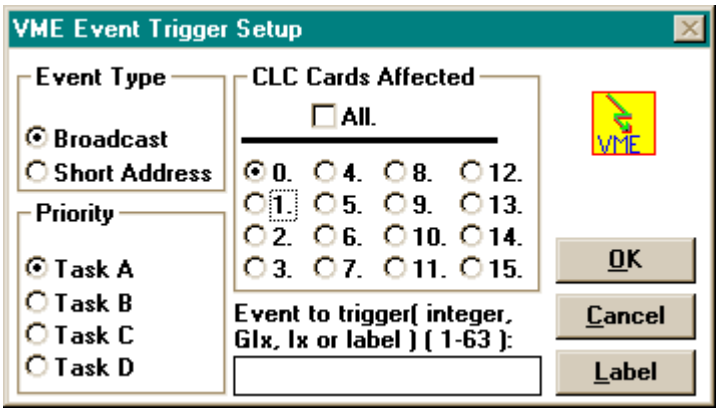
Data Format Conversion and Data Width Errors

Data is automatically converted to the data type of the specified target. For example, saving a CLC floating point variable to a VME memory location in I32 format automatically converts the float to a 32-bit integer. The conversion truncates the data if it exceeds the range of the target data type. It is the programmer's responsibility to check the data size and do any necessary data scaling before invoking the data transfer and associated automatic conversion.

A compile-time error will result if the VME source data width exceeds the selected data format width. In other words, don't attempt to move VME D16-width data into a U8 (unsigned integer) because the specified source data width is physically larger than the target width. For more information on the limitations of direct VME memory access ***see the CLC Start-Up Guide (CLC-V Backplane Communication section in the appendix, "CLC-V Hardware.")***

VMEEvt

The VMEEvt icon is used to trigger CLC events between cards on the same VME bus. In order for the VME event to trigger, the event function must be have been previously defined and enabled on the target card. You may enter an Event function by choosing the Add Event Function selection from Visual Motion's Edit menu. Events are enabled by an Event icon.



Selecting Broadcast Event Type broadcasts an event trigger to one or more CLC cards in a multi-card VME CLC system. Cards are individually selected, or the All checkbox can be used to select all cards at once. The Priority radio buttons select the triggering event's priority on the targeted cards, independent of the priority of the local task generating the trigger. Task A has the highest priority, Task D the lowest.

A Broadcast VME event is restricted to the first 63 events in the event table of the target card. (I.E. the event index must equate to less than 64.) The targeted CLC card(s) must have a local event function defined for the specified event index number and the event must be enabled on the target card(s). The event index is entered as an integer constant, variable (Ix), global variable (GIx), or an equivalent label. A User Defined Labels pop-up dialog box is available.

Selecting a Short Address event type writes to the VME bus short address space at a specific memory address (i.e., the CLC's' mailbox 4). For the event to occur, the event must be set up as a VME Short Address Event on the target card and the event must be enabled.

Only one VME Short Address Event on each card should be enabled at one time. If a program requires more than one Short Address event on a target card, it is the programmer's responsibility to insure that only one event is enabled at one time.

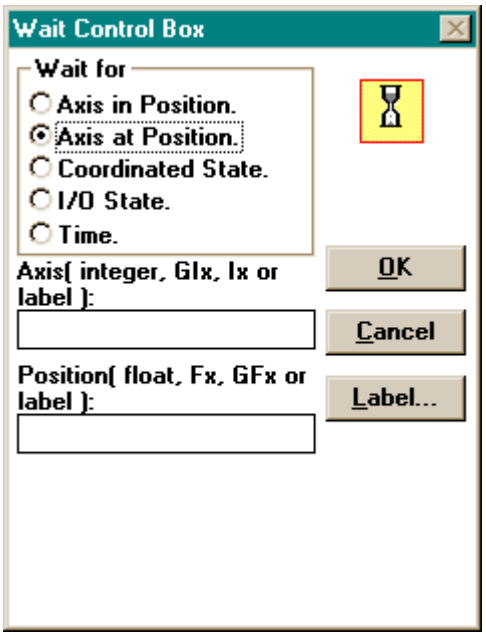
Wait

The Wait icon is used to hold the execution of the program flow at the Wait icon until a specified condition has been satisfied. The condition may be related to a single axis' position, time, I/O state, or path planner state (for coordinated motion). Placing a Wait icon on a task or subroutine workspace automatically displays a Wait Control Box dialog box.

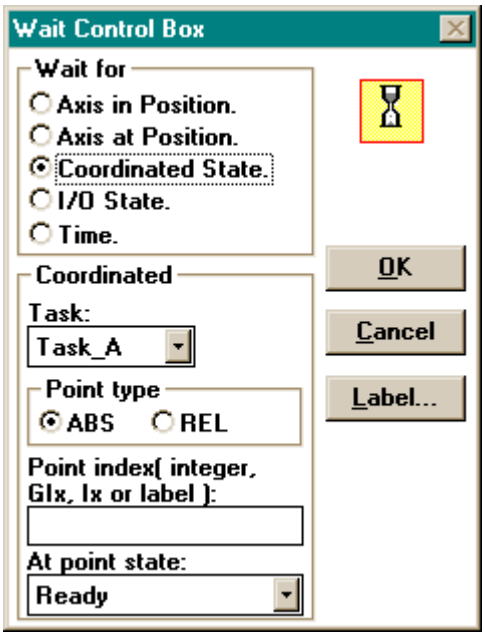
Axis in Position pauses program execution until the specified axis reaches the *in* position window of the associated drive. The axis may be specified by a valid integer constant variable (Ix) global variable (Gix) or an equivalent label.

If a "-1" is entered program execution will wait until all axes assigned to the task are within their respective position windows before continuing.

Axis at Position pauses program execution until the specified axis reaches a specified position for the associated drive.



Selecting **Coordinated State** pauses program execution and tests the state of the path planner for the specified point until the planner enters the selected processing state. Coordinated Waits are specific to each task. You can wait in one task for the coordinated state of another



The *At point state* pop-down menu provides the eight path planner test states listed below:

Segment Ready - Path planner has processed and queued the specified point.

Acceleration - Task's coordinated motion is accelerating into the segment ending at the specified point.

Slew (constant speed) - Task's coordinated motion is traversing the segment ending at the specified point.

Blending - Task's coordinated motion is traversing the blend segment calculated for the specified point.

Target Deceleration - Task's coordinated motion is in deceleration in the segment ending at the specified point.

Controlled Stop - Task's coordinated motion is decelerating on the segment ending at the specified point after a commanded stop.

Stopped - Task's coordinated motion is stopped after a commanded stop.

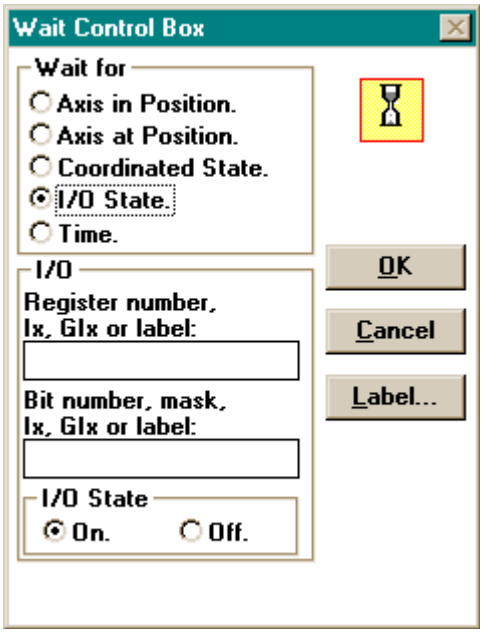
At Target - Task's coordinated motion is at the specified point.

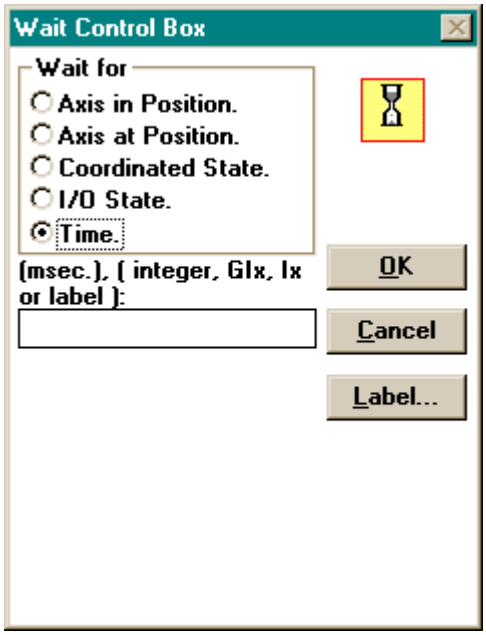
Done - Task's coordinated motion has completed for the specified point.

I/O State pauses program execution until the specified I/O condition for the specified I/O register is satisfied. The I/O register is specified by entering a valid I/O register ID number or equivalent label. A popup Register Labels dialog box is available

The contents of the specified register may be "anded" with a bit mask for the register contents. The bit mask allows "masking out" unwanted bits (by specifying "0" in the bit position in the mask), and permits the on/off condition to be effected by more than a single bit (by specifying "1" to enable the bit).A pop-up Bit Labels dialog box is available.

The I/O State radio buttons determine how the wait condition is satisfied. I/O State "on" (high or "1") requires that all the enabled bits are logical one. I/O State "Off" (low or "0") requires that all enabled bits are zero.





Time pauses program execution for a specified time delay. Enter the number of milliseconds in the Time Delay data entry box. The delay may be specified by an integer constant, variable (Ix), global variable (GIx), or an equivalent label. A pop up dialog box of User Defined Labels is available for both data entry boxes.

7 Text Language Programming

7.1 Overview

As an alternative to graphical Icon Programming, Visual Motion also provides a Text Language programming capability. Text Language permits writing CLC programs using Windows' Notepad or a similar ASCII text-only editor that does not use extended characters for formatting. The command syntax combines features of Basic and assembly language, with built-in extensions such as an IF-ELSE-ENDIF construct.

7.2 Directives

Directive statements provide CLC instructions that control the compiler, control the loading of programs between the CLC and Host, and control the manner in which the CLC manages the different portions of a user program.

Some of the directives are only used to initialize or configure the system. These directives are removed from the instruction stream and are not executed during the normal operation of a program.

EQU (Equate)

The EQU equate directive permits you to assign a symbolic name to a literal constant. Equating is most beneficial when using a constant repeatedly. An equated symbol name is global to all CLC tasks, subroutines and events within your application program. The name may have up to 20 ASCII characters. Remember that no white spaces are allowed within the name.

EQU statements should be placed at the very beginning of your program and must appear before the name is referenced in the program. Should you later need to change the value of the constant, only the single instance of the value in the EQU statement at the beginning of your program has to be modified.

Every time a program is compiled, the symbolic names in the program are replaced with their equated literal value.

Example:

```
RATIO1      equ 2.7348  ;a ratio between two coupled shafts
DONE  equ 0x1000  ;a bit mask for an I/O register
```

The compiler generates an error if a symbolic name is used in a program before it has been defined, or if the name has previously been used to define a variable, axis or identify a subroutine.

DEFINE

The DEFINE statement is used to assign a symbolic name to an integer or floating point variable in the corresponding variable table. Using a DEFINE permits you to use a mnemonic name instead of the "F[n]" or "I[n]" standard format for floats or integers. (See the section on Variables for details on the standard format.) A defined name may have up to 20 ASCII characters; no white spaces are allowed. A defined name is global to all CLC tasks, subroutines and events.

Unlike equates, defined names are downloaded and stored in separate variable tables in the CLC. Tables allow access to the variables by a user device through the CLC's serial communications port; or, if used, through a teach pendant.

Example:

```
DEFINE F05  Start_Veloc       ;a floating point user variable
DEFINE I03  Finished_Parts    ;an integer user variable
```

A DEFINE statement only associates a symbolic name with a variable. It does not initialize the variable to any value. If you need the variable to have an initial value, the value must be set using an expression, after the DEFINE statement and before the variable is used in the program.

Example:

```
Start_Veloc = 100       ; sets the defined variable F5 to 100
Finished_Parts = 5000   ; sets the defined variable I3 to 5000
```

As with the EQU statement, the compiler generates an error if a symbolic name is used in a program before it has been defined, or if the name has previously been used to define a variable, axis or identify a subroutine.

EVENT/START & EVENT/END

The EVENT/START and EVENT/END directives are used to tell the compiler that the program statements contained within the directives are an event function. Every EVENT/START statement must have an ending EVENT/END statement. Unlike a called subroutine, an event function should not have a RETURN statement at the end of the event function program

statements. Proper termination of an event is handled by the compiler and CLC multitasking executive.

Example:

```
Event1:          ;Mark to identify the beginning of the event program statements

EVENT/START
      .
      .          ;event program statements
      .
EVENT/END
```

TASK/START & TASK/END

The TASK/START and TASK/END instructions are used to identify the respective beginning and ending of program statements associated with a task. The task instructions do not perform any initialization, but are necessary indicators for the CLC compiler. Each TASK/START must have a "Task_(task letter):" mark before the TASK/START statement. Each block of program statements must have a corresponding TASK/END statement. Task statements should not be nested.

Example:

```
Task_A:          TASK/START  A ;Mark to identify the beginning of the task
statements
                 .
                 .
                 TASK/END    A ;task program statements
                 .
```

7.3 CLC Instructions

Instruction Format

The entire CLC statement line is limited to a maximum length of 80 characters including the optional label or mark, instruction, arguments, and comment.

White space may be used to format lines for easier readability. You may use tabs or spaces for white space. However, do not use white spaces in the middle of a label, mark, instruction, option or argument. If you need to separate a label or mark's characters for clarity, use the underscore character, "_".

Visual Motion Text Language program statements should be written using the following generalized format:

Example:

```
MARK:     INSTRUCTION/OPTION     ARG1, ARG2,...ARGn     ;COMMENT
```

where:

MARK is an optional user-assigned symbolic name immediately followed by a colon (:). A mark is typically used as an entry point to a series of program statements that begin with the line containing the mark.

INSTRUCTION is the CLC instruction mnemonic for the instruction.

OPTION is a modifier that may be required by the instruction and is separated from the preceding instruction by a forward slash (/) (known as a "virgule" if you care about typesetting terminology). White space is not allowed between the instruction and its modifier.

ARG1, ARG2, ...ARGn are arguments for the instruction. The number of arguments is specific to the instruction, ranging from none to several. If more than one argument is supplied to an instruction the arguments must be separated by commas. The argument(s) must be separated from the instruction by one or more white spaces.

COMMENT is a text string that you may enter to describe the purpose of an instruction statement. A comment must begin with a semi-colon (;). Comments are used only as an aid to understanding the program. When generating the executable program code, the CLC compiler ignores everything from the semicolon to the end of the line.

Writing Program Statements

Instructions must be separated from their arguments by at least one white space, either tabs or spaces. If more than one argument is used each argument must be separated by a comma ",".

An equivalent label may be used in place of a literal specification (such as a constant, variable or table entry), unless otherwise noted.

Using Comments

While you're in the process of writing a program you know what you intend the program statements to accomplish. You may not remember as well a month, or six months later. Using comments liberally always makes it easier to understand what the program is supposed to do. Multiple lines beginning with semicolon may be used to enter several lines of descriptive information about a subroutine, or as a "header" at the beginning of a program or major program section.

Example:

```
;HOMING SUBROUTINE
;This routine homes the x and y axes
;Expects as Inputs -
;     I/O register N = a value, bit n = another value
;Changes outputs -
;     I/O register N, bit n
;Changes system variables -
;     name1
;     name2
;etc.

SUBR01:     ;a mark must be used to identify the beginning of
            ;a subroutine's statements
      .
      .     ;subroutine program statements
      .
RETURN      ;return to the statement following the calling GoSub statement
```

The CLC instructions available for text language programming are listed in alphabetical order on the following pages. Each instruction is listed with a description, it's syntax and examples.

Braces ("{" and "}") are used to contain optional arguments, where more than a single argument may be supplied.

Example:

```
AXIS/EVENT  axis, event1 {, event2, event3, event4}
```

where:

axis and event1 are required, and event2, event3 and event4 are optional.

Labels generated by the DEFINE or EQU instructions may be used in place of constant or variable values.

AXIS/EVENT

Syntax:

AXIS/EVENT axis, event1 {, event2, event3, event4}

where:

argument	valid type(s)	range	description
axis	integer - constant - variable Ix,I[x] - global variable GIx,GI[x] - label	1 to the maximum number of valid axes	axis number
event1 event2 event3 event4	integer - constant - variable Ix,I[x] - global variable GIx,GI[x] - label	1 to the maximum number of valid events	event table index

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

The AXIS/EVENT instruction can enable up to four specified Repeating Position events for a single-axis, ELS, ratio, or velocity mode axis. The events are armed immediately, and an event function will execute each time the axis reaches the absolute position stored in the "a" field of the event specified in the event table.

Example:

```
        .
        .
  Axis_One    equ   1
  Cam_2 equ   15
  Cam_3 equ   16
        .
        .
  axis/event  Axis_One,Cam_2,Cam_3    ;activates repeating events
                                      ; Cam_2 and Cam_3 on Axis_One
        .
        .
```

AXIS/HOME

Syntax:

AXIS/HOME axis

where:

argument	valid type(s)	range	description
axis	integer - constant - variable Ix, I[x] - global variable GIx, GI[x] - label	1 to the maximum number of valid axes	axis number

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

AXIS/HOME signals the DDS to perform its drive-internal homing routine. CLC program execution pauses until a homing completed signal (or error message) is received from the DDS.

Before AXIS/HOME may be used, the appropriate homing parameters must be set in the DDS drive. Refer to the DDS manual for a description of the internal homing function and the required drive parameters and settings.

AXIS/HOME may be used with both coordinated and non-coordinated motion.

Example:

```
        .
        .
  define      I05,Axis_AX,
  Axis_BX     equ   4
  speed_BX    equ   100.00
  accel_BX    equ   250.00
        .
        .
        .
  axis/home   Axis_AX     ;homes the DDS-2 drive specified by integer variable 5
  axis/home   Axis_BX     ;homes the DDS-2 drive 4
```

AXIS/INITIALIZE

Syntax:

AXIS/INITIALIZE axis, speed, accel

where:

argument	valid type(s)	range	description
axis	integer - constant - variable Ix,I[x] - global variable GIx,GI[x] - label	1 to the maximum number of valid axes	axis number
speed	floating point - constant - variable Fx, F[x] - global variable GFx, GF[x] - label		initial speed
accel	floating point - constant - variable Fx, F[x] - global variable GFx, GF[x] - label		initial acceleration

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

AXIS/INITIALIZE is used for single axis non-coordinated motion. The instruction associates the specified axis with the user task containing the instruction. Initial speed and acceleration specified by variables may be modified by the user program.

Example:

```
        .
        .
        .
  define      F10,speed
  define      F11,accel
  define      I01,Grip_Event
  Grip_Event  equ   5
  TaskA_Ax1   equ   3
  speed       equ 100.0
  accel       equ 1000.0
        .
        .
        .
  axis/initialize   TA_Ax1, speed, accel
  axis/move         TA_Ax1, A, 100.0, Grip_Event
        .
        .
        .
```

AXIS/MOVE (Single Axis, Non-Coordinated)

Syntax:

AXIS/MOVE axis, mode, dist, {event1, event2, event3, event4}

where:

argument	valid type(s)	range	description
axis	integer - constant - variable Ix, I[x] - global variable GIx,GI[x] - label	1 to the maximum number of valid axes	axis number
mode	ASCII character	A for absolute R for relative	specifies the type of move
dist	floating point - constant - variable Fx,F[x], - global variable GFx, GF[x] - label		distance to move
event1 event2 event3 event4	integer - constant - variable Ix, I[x] - global variable GIx, GI[x] - label	1 to the maximum number of valid events	table index of first event table index of second event table index of third event table index of fourth event

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

AXIS/MOVE is used for single axis non-coordinated motion. Digital drives support this feature internally; no CLC path generation is needed, reducing the computational load on the path planner. Because the motion profile is generated within the drive, time-related events cannot be used with the AXIS/MOVE instruction.

Example:

```
speed       equ   100.0
accel       equ   1000.0
grip_event  equ   5
TaskC_Ax3   equ   7
F50   equ 100.0
      .
      .
      .
axis/initialize  TC_Ax3, speed, accel
axis/move        TC_Ax3, A, F50, Grip_Event
      .
      .
      .
```

AXIS/RATIO

Syntax:

AXIS/RATIO axis1, axis2, Mfactor, Sfactor

where:

argument	valid type(s)	range	description
axis1	integer - constant - variable Ix,I[x] - global variable GIx,GI[x] - label	1 to the maximum number of valid axes	master axis number
axis2	integer - constant - variable Ix,I[x] - global variable GIx,GI[x] - label	1 to the maximum number of valid axes	slave axis number
Mfactor	floating point - constant - variable Fx,F[x] - global variable GFx,GF[x] - label	32 bits	master axis factor
Sfactor	floating point - constant - variable Fx,F[x] - global variable GFx,GF[x] - label	32 bits	slave axis factor

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

AXIS/RATIO establishes a mathematical relationship between the number of revolutions of the specified slave axis that will result from the rotation of the specified master axis, according to the formula:

slave axis velocity = master axis velocity * (Sfactor/Mfactor)

The AXIS/RATIO command automatically updates the master factor parameter As.0031 and slave factor parameter As.0032. The separate factors are normalized before division so that the calculation maintains maximum precision with repeating decimals such as 2/3.

Specifying a negative number for one of the Factors produces reversed rotation in the slaved axis.

Multiple AXIS/RATIO statements may be used to link several slave axes to a single master.

A slaved axis must not be assigned to any task other than the task containing the master axis.

Follow ELS Master Axis

Optionally (through parameter A-0-0038), the slave axis may follow the velocity feedback of the master or be maintained in velocity mode instead of position mode. An ELS master may be used as the master axis, by choosing axis 0.

A ratio mode slave can now follow the ELS master (virtual or real). On a system without ELS drives, all slaves can follow the ELS master so that there is no lag between slaves. To follow the ELS master, enter 0 for the master axis in the axis_setup icon or axis/setup text language command. If no ELS slave axes are present, it is necessary to set the following ELS parameters using parameter/init statements: C-0-0150, C-0-0151, C-0-0152, C-0-1000, C-0-1001.

Example:

```
Task_A:
task/start  A                ;start of code for task A
task/axis   1, 2         ;assign axes for task A
axis/ratio  1, 3, 4.0, -5.0   ;axis 3 linked to axis 1 at 5/4
      .
      .
      .
;Operation results in 5 revolutions of axis 3 (in the reverse direction) for each
;4 revolutions of axis 1.
```

AXIS/SPINDLE (Continuous Velocity Mode)

Syntax:

AXIS/SPINDLE axis, rpm

where:

argument	valid type(s)	range	description
axis	integer - constant - variable Ix,I[x] - global variable GIx,GI[x] - label	0 to the maximum number of valid axes	axis number
rpm	floating point - constant - variable Fx,F[x] - global variable GFx,GF[x] - label	32 bits	revolution per minute

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

The AXIS/SPINDLE instruction tells the specified axis (and consequently the DDS drive) to run at a constant velocity. The acceleration ramp profile is generated within the drive, according to the previously set drive acceleration parameters. The sign of the speed argument determines the direction of rotation.

This instruction also enables the axis to go; the ‘Axis/Start’ command need not be used.

The current speed of the axis can be monitored by reading the axis speed parameter from the drive using the PARAMETER/GET instruction.

Example:

```
        .
        .
        .
  speed       equ   100.0
  accel       equ   1000.0
  TA_Ax1      equ   4
        .
        .
        .
  axis/initialize   TA_Ax1, speed, accel
  axis/spindle      TA_Ax1, -22.2
        .
        .
        .
```

AXIS/START

Syntax:

AXIS/START axis

where:

argument	valid type(s)	range	description
axis	integer - constant - variable Ix,I[x] - global variable GIx,GI[x] - label	0 to the maximum number of valid axes	axis number

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

AXIS/START begins motion for the selected axis. The instruction signals the DDS drive(s) to begin motion toward the target position set by the AXIS/MOVE command. The axis must be programmed for non-coordinated motion. (E.G., single-axis or velocity mode.)

Once an axis has been started, each new AXIS/MOVE instruction begins a new movement, without requiring an AXIS/START.

If the specified axis is set to "-1", the instruction begins motion for all single-axis and velocity mode axes programmed in the task. A program may set target positions for several axes using multiple AXIS/MOVE commands, then begin motion on all axes at the same time with a single "-1" argument to AXIS/START.

Example:

```
        .
        .
        .
  TaskA_AX1   equ   1     ;use drive 1
  TaskA_AX2   equ   3     ;use drive 3
  TaskA_AX3   equ   5     ;use drive 5
        .
        .
        .
  axis/move   TaskA_AX1, A, F50       ;set position for drive 1
  axis/move   TaskA_AX2, A, F55       ;set position for drive 3
  axis/move   TaskA_AX3, R, F60       ;set position for drive 5
        .
        .
        .
  axis/start  -1          ;start motion, all axes
        .
        .
        .
```

AXIS/STOP

Syntax:

AXIS/STOP axis

where:

argument	valid type(s)	range	description
axis	integer - constant - variable Ix,I[x] - global variable GIx,GI[x] - label	0 to the maximum number of valid axes	axis number

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

AXIS/STOP halts motion for the selected axis. The instruction signals the DDS drive(s) to decelerate the drive to zero velocity using the rate set in the drive's deceleration parameter. The axis must be programmed for non-coordinated motion. (E.G., single-axis or velocity mode.)

Once an axis has been stopped, an AXIS/MOVE instruction does not effect movement of the axis until an AXIS/START instruction is executed.

If the specified axis is set to "-1", the instruction disables motion for all single-axis and velocity mode axes programmed in the task. A typical use is to synchronize motion on several axes, or to stop all motion on an I/O or error condition.

Example:

```
        .
        .
        .
  TaskA_AX1   =     1
  TaskA_AX2   =     3
  TaskA_AX3   =     5
        .
        .
        .
  axis/move   TaskA_AX1, A, F50       ;set position for drive 1
  axis/move   TaskA_AX2, A, F55       ;set position for drive 3
  axis/move   TaskA_AX3, R, F60       ;set position for drive 5
  axis/start  -1                      ;enables all drives
        .
        .
        .
  axis/stop   -1          ;stops motion on all axes
        .
        .
        .
```

AXIS/WAIT (Axis Wait For In-Position)

Syntax:

AXIS/WAIT axis

where:

argument	valid type(s)	range	description
axis	integer - constant - variable Ix,I[x] - global variable GIx,GI[x] - label	0 to the maximum number of valid axes	axis number

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

AXIS/WAIT suspends task execution until the DDS drive associated with the specified axis signals the CLC that the axis is in position and zero velocity. The drive's in position and zero velocity values are set by the drive's Position Window and Zero Velocity parameters.

If the specified axis is set to "-1", the instruction waits until all axes in the task are positioned. A typical use is to synchronize motion on several axes.

If the axis position is never achieved the task remains waiting, suspended indefinitely.

Example:

```
        .
        .
        .
  event1             equ   10
  Turn_Gripper_On    equ   12
  TA_Ax1             equ   3
  define             F10,speed
  define             F11,accel
  speed              equ   00.0
  accel              equ   1000.0
        .
        .
        .
  Axis/Initialize    TA_Ax1,speed,accel
  Axis/Move          TA_Ax1,R,100.0,Event1,Turn_Gripper_On
  Axis/Wait          TA_Ax1
        .
        .
        .
```

CALL

Syntax:

retval=CALL function_label, arg1, arg2, ...argN

where:

argument	valid type(s)	range	description
function_label	label	any valid function label	name of function
arg1 ...argN	Fx, Ix, GFx, GIx, integer label, float label or point label		values passed to function
retval (optional)	Fx, Ix, GFx, GIx, integer label or point label		variable to recieve return value of function

Description:

This instruction calls the function ‘function_label’ with the arguments specified, and optionally returns a value.

Example:

```
;  Main task calling subroutine and returning value
Task_A:     TASK/START A
            I1 = CALL         sub, 5, 10, 20
            TASK/END A
;  Subroutine to multiply input values
sub:  FUNCTION/START U
            FUNCTION/ARG      COUNT1, I, 1, 300
            FUNCTION/ARG      COUNT2, I, 1, 300
            FUNCTION/ARG      COUNT3, I, 1, 300
            LOCAL/VARIABLE    TAB, I
            TAB = (COUNT1 * COUNT2) * COUNT3
            FUNCTION/END TAB
```

CAM/ACTIVATE

Syntax:

CAM/ACTIVATE axis, cam, M, N, H, L

where:

argument	valid type(s)	range	description
axis	integer - constant - variable Ix,I[x] - global variable GIx, GI[x] - label	1 to the maximum number of valid axes	axis number
cam	integer - constant - variable Ix,I[x] - global variable GIx,GI[x] - label	1 to the maximum number of valid cam tables	specify cam table ID
M, N, H, L	floating point - constant - variable Fx,F[x] - global variable GFx,GF[x] - label	32 bits	define coefficients in cam equation. If L is not specified, it is assumed L=0.

Description:

The CAM/ACTIVATE instruction is used to associate a cam to an axis and to supply coefficents for using the cam. For further information on cam theory of operation, refer to the Cam Icon in the Icon Programming chapter.

CAM/ADJUST

Syntax:

CAM/ADJUST axis, mode, type, degrees

where:

argument	valid type(s)	range	description
axis	constant, integer, GIx, Ix, label	any valid axis number	axis to adjust
mode	integer	1 - master phase adjust 2 - slave phase adjust	selects master or slave (Mph or Sph in equation)
type	integer	1 - absolute 2 - incremental 3 - continuous + 4 - continuous -	
degrees/ percent	Fx, GFx, GIx, integer or label		phase adjust target value

Description:

This instruction selects which phase adjust to perform and starts the phase adjust. With type=1, it functions the same as the ELS/ADJUST instruction. ELS/ADJUST functions the same as before for compatibility with existing programs.

There are two phase offset values for a cam axis: a master phase adjust, and a slave phase adjust. The master phase adjust shifts the position in the cam table relative to the master position. The slave phase adjust shifts the position of the slave axis. Since it is not related to the shape of the cam, the slave phase adjust is not multiplied by any of the cam factors.

The CLC can perform only one phase adjust at a time. This instruction will have no effect until the previous phase adjust is complete if the phase adjust type is different. Bit 4 in the axis status register is set to (0) when a phase offset is in progress, and (1) if the phase offset is complete.

Example:

```
.
.
CAM/ADJUST  1,2,1,35
.
.
```

CAM/BUILD

Syntax:

CAM/BUILD cam_number, option, start_point, end_point, wait

where:

argument	valid type(s)	range	description
cam_number	integer - constant	1-8 = cam stored on CLC	integer or constant containing number of cam to be built
option	integer - constant	1= use pcam utility 2= use spline utility	cam generation option
start_point	integer - constant		starting index in the ABS point table to use for cam generation
end_point:	integer - constant		end index in the ABS point table used for cam generation
wait:	constant	0 or 1	**0**= don't wait for completion of build instruction **1**= wait for the cam to be ready for activation

Description:

The CAM/BUILD command builds a cam internally on the CLC and stores it on the CLC in the cam number indicated. The CLC's ABS point table is used to build the cam. The ABS point elements may be changed from within the program using CALC statements. Changes in the point table do not affect the cam until the CAM/BUILD command is executed.

For the pcam option, the X elements of the point table are the master positions, and the Y elements are the corresponding slave positions. The CLC builds a jerk-limited position profile between these points.

The cam generation may take one second or longer. The 'wait' option can be set to (0) to exit this instruction immediately and keep executing instructions while the cam is being built.

The CAM/STATUS instruction can be used to check if a cam is ready for activation. If the 'wait' option is set to (1), the program flow will be stopped in this instruction until the cam is ready for activation.

The CAM/BUILD command can be used to store a cam to an inactive location on the CLC. After the cam has been built, the CAM/ACTIVATE command can activate it for an axis.

Notes

Because the cam is stored in nonvolatile memory on the CLC, it is not necessary to execute this command each time through the program. A flag variable can be set and checked the next time through the program to avoid long delays when starting the program. For on-line changes, a register bit or variable should be checked each time through the program loop to avoid continually generating the cam, which consumes CLC resources and can slow down the program.

It is necessary to size the point table at compile-time to allow enough points for the profiles that will be needed.

Errors will be issued when:

The selected cam is currently active for any axis.

The point range exceeds the bounds of the point table.

Less than two points are defined.

The cam number is not valid (out of range or drive is not configured).

An error occurred when sending the cam to the drive.

When using PCAM option, and the first x position isn't 0 and the last x position isn't 360.

The x position exceeds the modulo of the master.

CAM_INDEX Command

Syntax:

CAM/INDEX

where:

argument	valid type(s)	range	description
CAM_INDEX	cam id, floatBlock, intBlock		CAM indexer instruction
VAR/INIT	floatBlock, feedlength, feedStart, feedStop, diameter, eventTime, accel, decel, dwell, preDwell		Sets up float variables at compile time
VAR/INIT	intBlock, feedStopType(Bit 0) feedProfileType		Sets up integer variables at compile

CAM/STATUS Command

Syntax:

Refer to IF CAM/STATUS Command

Description:

After a cam download or after the CAM/BUILD command, it takes some time to calculate and store the new cam. When a cam is activated on the CLC, it does not run until the cam is at the end of its cycle.

This command sets a task's status word with a logical result of the cam status being check for.

Status: integer of status being checked for

0= no valid cam is stored
1= cam is being calculated
2= cam is being sent to drive (drive cams only)
3= cam is ready for activation
4= cam is active and running (CLC cams: for any axis)

Errors will be issued when:

The cam number is not valid (out of range or drive is not configured).

CAPTURE/ENABLE

Syntax:

CAPTURE/ENABLE axis, probe, event

where:

argument	valid type(s)	range	description
axis	integer - constant - variable Ix,I[x] - global variable GIx, GI[x] - label	1 to the maximum number of valid axes	axis number
probe	integer - constant - variable Ix,I[x] - global variable GIx,GI[x] - label	1 to 4	specify DDS drive probe and edge transition
event	integer - constant - variable Ix,I[x] - global variable GIx,GI[x] - label	1 to the maximum number of valid events	enable specified event and trigger on probe transition

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

The CAPTURE/ENABLE instruction enables a feedback capture event for an axis. Each axis can have up to four active capture events (one for each probe edge).

Specifying "0" for the event number disables capture for the specified probe transition.

Probe specifies which of the two probe inputs that will be enabled or disabled, and a transition direction (positive or negative going) for the input. An error is issued if the probe and edge were not selected as a trigger using a CAPTURE/SETUP command.

1 = Probe 1 positive transition 2 = Probe 1 negative transition
3 = Probe 2 positive transition 4 = Probe 2 negative transition

When the drive detects the selected transition on its probe input, the CLC transfers the probe position into the event table element "a" (i.e., EVT[n].a) specified by the event number, and executes the optional associated event function.

The CALC statement can be used to read the position from the event table and perform any necessary calculations or adjustments. The probe position is also placed in a CLC axis parameter.

This instruction does not enable a repeating event. A CAPTURE/ENABLE statement must be executed for each new edge transition to be captured.

Example:

```
        .
  Capture/Setup     Axis_1, In_Place              ;setup for trigger
        .
        .
  Capture/Event     Axis_1, In_Place, Mark_edge   ;capture position on trigger
        .
```

CAPTURE/SETUP

Syntax:

CAPTURE/SETUP axis, trigger1

where:

argument	valid type(s)	range	description
axis	integer - constant - variable Ix,I[x] - global variable GIx, GI[x] - label	1 to the maximum number of valid	axis number
trigger1	integer - constant - variable Ix,I[x] - global variable GIx,GI[x] - label	1 to 4	specifies the first DDS probe and trigger edge
trigger2	integer - constant - variable Ix,I[x] - global variable GIx,GI[x] - label	1 to 4	specifies the second DDS probe and trigger edge

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

CAPTURE/SETUP is executed only at program activation. The instruction configures a CLC axis and associated DDS-2 drive to use the internal position feedback capture capability of the drive. One or two triggers can be selected, each may be triggered by either a low-to-high or high-to-low transition of either probe input.

The trigger selects a probe input edge to be enabled in the drive and allocated to SERCOS real-time bit 1. The CLC also allocates space in the drive's cyclic data for the captured probe position. Trigger1 may be specified by:

1= Probe 1 positive transition 2= Probe 1 negative transition
3= Probe 2 positive transition 4= Probe 2 negative transition

Example:

```
      .
Capture/Setup     Axis_1, In_Place              ;setup for trigger
      .
      .
Capture/Event     Axis_1, In_Place, Mark_edge   ;capture position on trigger
      .
```

DATA/SIZE (Configure Memory Allocations)

Syntax:

DATA/SIZE Integers, Floats, Apoints, Rpoints, Events, Zones

Where:

argument	valid type(s)	range	description
Integers	integer - constant	0 to available CLC memory	task memory allocated for Integer variables
Floats	same as above	same as above	task memory allocated for Floating Point variables
Apoints	same as above	same as above	task memory allocated for Absolute Point Table entries
Rpoints	same as above	same as above	task memory allocated for Relative Point Table entries
Events	same as above	same as above	task memory allocated for Event Table entries
Zones	same as above	same as above	task memory allocated for Zone Table entries

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

The DATA/SIZE instruction is used within each task to specify the amount of memory to be allocated to each data type required by the task. The total CLC memory allocated for a program is the sum of the allocations for each data type, in each of the four tasks. Once allocated, memory is global and may be accessed by all four tasks.

The DATA/SIZE command can only be used within the four main tasks. It must not be used within a subroutine or event function.

Example:

```
;Task A
Int_NumA    equ   20
FP_NumA     equ   20
ABS_NumA    equ   30
REL_NumA    equ   20
      .
      .
      .
;Task B
ABS_NumB    equ   10
REL_NumB    equ   10
EVT_NumB    equ   10
      .
      .
      .
Task_A:
Task/Start  A
Data/Size   Int_NumA, FP_NumA, ABS_NumA, REL_NumA, 0
      .
      .           ;program code
      .
Task/End    A
      .
      .
      .
Task_B:
Task/Start  B
Data/Size   0, 0, ABS_NumB, REL_NumB, EVT_NumB
      .
      .           ;program code
      .
Task/End    B
      .
      .
      .
```

Each integer or floating point entry requires 4 bytes of CLC memory.

Each absolute or relative point table entry requires 44 bytes of CLC memory.

Each event table entry requires 120 bytes of CLC memory.

Each zone table entry requires 28 bytes of CLC memory.

The CLC memory allocated for the program is:

Data Type	Task A	Task B	Task C	Task D	Total
integer	20	0	0	0	20
floating point	20	0	0	0	20
absolute points	30	10	0	0	40
relative points	20	10	0	0	30
events	0	10	0	0	10
zones	0	0	0	0	0
memory required in bytes	2360	2080	0	0	4440

DEFINE (Define Name For Variable)

Syntax:

DEFINE variable, label

where:

argument	valid type(s)	range	description
variable	integer - variable Ix or Fx - global variable GIx or GFx	a valid variable	
label	ASCII	20 characters maximum	

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

DEFINE assigns a name to an integer or floating point variable or global variable.

White spaces (spaces or tabs) are not allowed within the character string.

The assignment must occur in the program before the defined variable is used.

The compiler generates an error if a name is used before it is defined; or if a subroutine, or another variable or constant has been previously assigned the same name.

Example:

```
        .
        .
        .
  True        equ   1
  False       equ   0
  Define      F10, Rate_Of_Change     ; Names F10
  Define      I22, Logical_State      ; Names I22
        .
        .
        .
  Task_A:
  Task/Start  A
  Rate_Of_Change    =     COS(F10)/2.0      ; Set the rate
  Logical_State     =     True              ; Set state true
        .
        .     ;program statements
        .
  Task/End    A
```

DELAY (Suspend Task Execution)

Syntax:

DELAY time

where:

argument	valid type(s)	range	description
time	integer - constant - variable I[x] - global variable GI[x] - label	1 to 360000 (32 bit maximum)	time delay in milliseconds

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

When the program execution flow encounters the DELAY statement, the task issuing the instruction is suspended for the specified period of time. After the delay, task execution resumes with the next statement.

Example:

```
        .
        .
        .
  delay       10          ; Delay 10 milliseconds
        .
        .
        .
```

ELS/ADJUST (Adjust ELS Axis)

Syntax:

ELS/ADJUST slave_axis, fine_adjust, adjust_type
or,
ELS/ADJUST slave_axis, phase_offset, adjust_type
where:

argument	valid type(s)	range	description
slave_axis	integer - constant - variable Ix,I[x] - global variable GIx,GI[x] - label	1 to the maximum number of valid	axis number of slave
fine_adjust	floating point - constant - variable Fx,F[x] - global variable GFx,GF[x] - label	-100% to +300%	new value of velocity sync. fine ratio adjust
phase_offset	floating point - constant - variable Fx,F[x] - global variable GFx,GF[x] - label	0° to +360°	new value of phase sync. phase offset
adjust_type	absolute incremental continuous + continuous -		

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

For ELS drives in velocity synchronous mode, this instruction sets the Ratio Fine Adjust parameter (P-0-83) on the drive. This parameter is a percent from -100 to 300, allowing an adjustment from stopping the slave to four times the parametric ratio (S-0-236 and S-0-237, set in parameter mode).

For ELS drives in phase synchronous mode, this instruction adjusts the phase offset in degrees. The drive will shift in position relative to the master based on the value in this constant or float variable.

If either parameter needs to read, the parameter transfer instruction can be used. This is useful when a variable must be set to an initial phase offset value. The parameter for phase offset is As.151, where s is the slave axis number.

Example:

```
DIST        EQU   20.0          ; Sets equate
      .
      .
TASK_B:TASK/START B
ELS/INIT    1, 1, 2, 1, 1       ; ELS axis initialization
ELS/MODE    1, 1                ; Slave synchronized to master via
                  ; ELS
ELS/ADJUST  1, -100             ; Fine adjust for velocity sync.
      .
      .
      .
```

ELS/DECEL (Set Master Decel)

Syntax:

ELS/DECEL rate

where:

argument	valid type(s)	range	description
rate	floating point - constant - variable Fx,F[x] - global variable GFx,GF[x] - label	32 bits	sets the ELS master axis deceleration rate in radians

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

This command sets the deceleration rate for the ELS virtual master in radians/sec^2.

Example:

```
DEFINE      F1, RATE          ; Defines label 'RATE'

TASK_A:TASK/START A
       .
       .
ELS/INIT    1, 1, 2, 1, 2     ; ELS axis initialization
       .
       .
ELS/DECEL   RATE              ; Sets deceleration rate
       .
       .
TASK/END    A
```

ELS/INIT (Initialize ELS Axes)

Syntax:

ELS/INIT els_type, slave_axis, master_axis, encoder_type, sync_type

where:

argument	valid type(s)	range	description
els_type	integer - constant - label	1 to 4	1= Virtual master through SERCOS (default) 2= Real master to each slave 3= Real master through SERCOS 4= Follow axis feedback
slave_axis	same as above	1 to maximum number of valid axes	the axis synchronized to the selected master
master_axis	same as above	1 to maximum number of valid axes	If els_type is set to 1 or 2, this value is ignored and the master axis is set to 0. If els_type is set to 3, master_axis must be set to the ID of the drive connected to the external feedback. Only drives with an external feedback option (such as the DDS) may be selected. If els_type is set to 4, master_axis must be set to the drive the slave axes are to follow. Any SERCOS drive may be selected.
encoder_type	same as above	1 or 2	1= Primary encoder (default) 2= Secondary encoder This value selects the encoder that is used when els_type is set to 3. ("Real Master Through SERCOS")
sync_type	same as above	1, 2 or 3	1= Velocity synchronization 2= Phase synchronization 3= Cam Synchronization This value sets the type of synchronization between the master and the slave.

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

In a CLC System there can be one ELS master, either virtual or real. The CLC executes the ELS/INIT command at program activation to automatically configure the master and slave drives. This command may be placed in any task and allows some of the axes in a system to be in manual mode while the others are following the master.

The four ELS master axis types are:

Virtual master through SERCOS (default). The CLC generates the master position to be followed by ELS slave drives. The virtual master uses the programmed velocity, acceleration, deceleration and E-stop deceleration rates.

Real master to each slave. Each slave drive uses its own external feedback, which can be daisy-chained to several drives to provide a common position reference.

Real master through SERCOS. The CLC reads the position from an external encoder connected to any drive and sends it through SERCOS to all the slaves as the master position.

Follow axis feedback. The CLC reads the feedback position from the master axis and sends it through SERCOS as the master position.

Example:

```
SLAVE_AXIS  EQU   1                              ; Sets equate
MASTER_AXIS EQU   2                              ;      "

TASK_C:TASK/START C
DATA/SIZE   5, 5, 5, 5, 5, 0                     ; Allocates memory for task C
ELS/INIT    4, SLAVE_AXIS, MASTER_AXIS, 1, 2     ; ELS axis initialization

      .
      .
TASK/END    C
```

ELS/MODE (Set ELS Axis Mode)

Syntax:

ELS/MODE slave_axis, mode

where:

argument	valid type(s)	range	description
slave_axis	integer - constant Ix, I[x],GIx,GI[x] - label	1 to the maximum number of valid axes	axis number of slave
mode	integer - constant Ix, I[x],GIx,GI[x] - label	1 2 3	Single-axis mode Slave synchronized to master via ELS Velocity

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

This instruction sets the operating mode for an axis. It switches the drive between single-axis mode and synchronous mode. If slave_axis is set to (-1), the mode is switched for all slave axes in the task.

To enable synchronization, set "mode" to 2. The drive is synchronized to the master in phase synchronous or velocity synchronous mode. When switching into phase synchronous mode, the Phase Offset parameter is automatically set to (slave position - master position).

To disable synchronization at any time during the program, set "mode" to 1. The slave switches to single-axis mode and is halted. To run the slave in single-axis mode, the single-axis instructions may be used.

Set “mode” to 3 to switch an axis that is configured for Single Axis Mode into Velocity Mode.

Example:

```
TASK_D:TASK/START D
       .
       .
ELS/INIT    1, 1, 2, 1, 1     ; ELS axis initialization
ELS/MODE    1, 1              ; Single Axis mode
ELS/DECEL   F1                ; Sets deceleration rate
```

ELS/STOP

Syntax:

ELS/STOP position

where:

argument	valid type(s)	range	description
position	constant - variable Fx, F[x], - global variable Gfx,Gf[x] - label	range of floating point numbers	position in which the virtual master will stop

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

This command specifies the target position in which the master velocity will come to a stop. The deceleration value will reflect on when the master velocity begins to stop.

Example:

```
.
.
ELS/STOP 0
.
.
```

EVENT/DONE (Signal Event Completed)

Syntax:

EVENT/DONE event

where:

argument	valid type(s)	range	description
event	integer - constant - variable Ix,I[x] - global variable GIx,GI[x] - label	1 to the maximum number of valid events	disable the event

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

The EVENT/DONE instruction changes the status of an active event. Time based events are taken out of the event timer queue, distance based events are made inactive if in the event queue or pending. EVENT/DONE is primarily used to disable repeating timer events, however, it immediately disables any event.

Executing an EVENT/DONE has no effect on an inactive event.

Example:

```
        .
        .
        .
  time  equ   1                        ; use 0 for time
  dist  equ   1                        ; use 1 for dist
        .
        .
        .
  Task_A:
  Task/Start  A
  EVT[3].t = time                     ; Set event 3 type to time
  EVT[3].a = 10                       ; Set event 3 time to 10ms
  EVT[3].m = "Motion Event Message"   ; Set the message for event 3
  EVT[3].f =  Gripper                 ; Set the subroutine mark for event 3
      .
      .                               ; Program code
      .
  Event/trigger     3                 ; Trigger event 3
  Event/wait        3                 ; Wait until done
      .
      .                               ; More program code
      .
  Event/done     3                    ; event done
  Task/End    A
  Gripper:                            ; Define event 0 function subroutine
  Event/Start                         ; Mark the beginning of the event
        plc/set     1, 2              ; Turn on gripper
  Event/End                           ; Mark the end of the event
        .
        .
        .
```

EVENT/END (Mark End of Event)

Syntax:

EVENT/END

Description:

Executing the EVENT/END instruction returns program execution to the invoking program, enabling normal cycling.

This instruction is required by the complier to indicate the end of an event function, and as an interrupt return for the CLC run-time executive.

Example:

```
        .
        .
        .
  time        equ   1                          ; Set equate
  dist        equ   3                          ; Set equate
        .
        .
        .
  Task_A:
  Task/Start  A
  EVT[3].t    =     time                       ; Set the event type
  EVT[3].a    =     10                         ; Set the event time (ms)
  EVT[3].m    =     "Motion Event Message"     ; Set the message for event
  EVT[3].f    =     Grip                       ; Set the function for event
        .
        .
        .
  Event/Trigger     3                          ; Trigger event 3
  Event/Wait        3                          ; Wait until done
        .
        .                                      ;additional program statements
        .
  Task/End    A
  Grip: Event/Start                            ; Define event
        PLC/Set     1, 2                       ; Turn on gripper
        Event/End                              ; End of event
        .
        .
        .
```

EVENT/START (Start of Event function)

Syntax:

EVENT/START

Description:

EVENT/START is a compiler directive used within a program to indicate the beginning of an event function.

Example:

```
time              equ   1                       ; Set equate
dist              equ   3                       ; Set equate
      .
      .
      .
Task_A:
Task/Start        A
EVT[3].t          =     time                    ; Set the event type
EVT[3].a          =     10                      ; Set the event time (ms)
EVT[3].m          =     "Motion Event Message"  ; Set the message for event
EVT[3].f          =     Grip                    ; Set the function for event
Event/Trigger     3                             ; Trigger event 3
Event/Wait        3                             ; Wait until done
      .
      .     ;program statements
      .
Task/End          A
      .
      .
      .
Grip: Event/Start                               ; Define event
      PLC/Set     1, 2                          ; Turn on gripper
      Event/End                                 ; End of event
      .
      .
      .
```

EVENT/TRIGGER (Trigger a Task Event)

Syntax:

EVENT/TRIGGER event

where:

argument	valid type(s)	range	description
event	integer - constant - variable Ix, I[x] - global variable GIx, GI[x] - label	1 to the maximum number of valid events	start event timer

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

EVENT/TRIGGER starts a repeating timer event. The subroutine specified by the event is repeatedly called at the rate specified in the event table entry. Task A's events have the highest priority.

EVENT/TRIGGER has no effect on repeating timer events that are already active.

An EVENT/TRIGGERed subroutine does not execute and enters a wait state if the task's event queue is full.

Caution: The execution time of the event function must not exceed the specified event repeat time. If the event function does not return before the event is re-triggered, the multitasking executive will execute the event function again, as soon as the event function returns. Program task execution will be blocked. Remember that ALL pending events, for a task, will complete before any task resumes execution.

Example:

```
time  equ   1                       ; Set equate
dist  equ   1                       ; Set equate
      .
      .
      .
Task_A:
Task/Start  A
EVT[3].t = time                     ; Set the event type
EVT[3].a = 10                       ; Set the event time (ms)
EVT[3].m = "Motion Event Message"   ; Set the message for event
EVT[3].f = Gripper                  ; Set the function for event
      .
      .                             ; Program code
      .
Event/trigger     3                 ; Trigger event 3
Event/wait        3                 ; Wait until done
      .
      .                             ; Yet more program code
      .
Task/End    A
      .
      .
      .
Gripper:                            ; Define event 0 function
Event/Start
      plc/set     2, 6              ; Turn on gripper
Event/End
```

EVENT/WAIT (Pause Task for Event Done Signal)

Syntax:

EVENT/WAIT event

where:

argument	valid type(s)	range	description
event	integer - constant - variable Ix,I[x] - global variable GIx,GI[x] - label	1 to the maximum number of valid event	pause task for event

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

Event/Wait suspends the task it is issued from until an event done instruction for the event is executed.

A CLC run-time error results if this instruction is directed toward an inactive event.

Example:

```
time  equ   1                       ; Set "time" to a constant
dist  equ   1                       ; Set "dist" to a constant
      .
      .
      .
Task_A:
Task/Start  A
EVT[3].t = time                     ; Set the event 3 type to time
EVT[3].a = 10                       ; Set the event 3 time (ms)
EVT[3].m = "Motion Event Message"   ; Set the message for event 3
EVT[3].f = Gripper                  ; Set the subroutine to use for event 3
    .
    .                               ; task program statements
    .
Event/trigger     3                 ; trigger event 3
Event/wait        3                 ; pause task until done
    .
    .                               ; additional program statements
    .
Task/End    A
Gripper:                            ; event 3 subroutine called from another task
Event/Start
      plc/set     2, 6              ; turn on gripper, I/O register 2, bit 6
Event/End
```

FUNCTION/ARG

Syntax:

FUNCTION/ARG label, type, min_value, max_value

where:

argument	valid type(s)	range	description
label	ASCII string	1 to 20 characters	name of argument
type	ASCII character	'I' = integer 'F' = floating point 'ABS'= absolute point index 'REL'= relative point index	specifies type of access
min_value	constant or label		optional minimum value of argument
max_value	constant or label		optional maximum value of argument

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

These instructions define the arguments passed to a function. Each argument is defined on a separate line immediately following the FUNCTION/START. The arguments must be defined in the order in which they are to appear in the CALL instruction or in the sequence table. An optional minimum and maximum value are avaliable to the user interface for validation. A maximum of 5 function arguments may be used. A total of 16 local variables

(LOCAL/VARIABLE, FUNCTION/ARG) may be used per function. Function argument labels are accessible from the teach pendant in a sequencer.

Example:

```
.
.
;     Function:   plc_float
;     Reads value from a 16-bit PLC register and return a scaled value

plc_float: FUNCTION/START     U                 ; access from teach pendant
      FUNCTION/ARG      regnum, I, 150, 160     ; register number of PLC register
      FUNCTION/ARG      scaler, F, 1, 100000    ; scaling value from PLC to float
      LOCAL/VARIABLE    itemp, I                ; local temporary variable
      LOCAL/VARIABLE    retval, I               ; local temporary variable
      PLC/READ          regnum,  1, itemp       ; read register from PLC
      retval = itemp * scaler                   ; scale to floating point
      FUNCTION/END retval                       ; return with retval
```

FUNCTION/END

Syntax:

FUNCTION/END value (optional)

where:

argument	valid type(s)	range	description
value (optional)	constant, floating point or integer label		return value from function

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

This instruction defines the end of a function. If no value is included, the function returns 0. The function may return only one value, which may be a floating point or integer. The value is returned in the variable specified in the CALL instruction.

Example:

```
.
.
      FUNCTION/END retval                    ; return with retval
```

FUNCTION/START

Syntax:

function_label: FUNCTION/START access_type

where:

argument	valid type(s)	range	description
function_label	label	any valid function label	name of function
access_type	ASCII character	'U' = Accessible in user function sequencer list 'N'= Not accessible	specifies type of access

Description:

This instruction indicates the start of a function. The arguments and return value are declared in the FUNCTION/ARG, LOCAL/VARIABLE and FUNCTION/END commands. If access_type is not included, the function is not accessible in the user function list.

Example:

```
;  Main task calling subroutine and returning value

Task_A:     TASK/START A
            I1 = CALL         sub, 5, 10, 20
            TASK/END A

;  Subroutine to multiply input values

sub:  FUNCTION/START U
            FUNCTION/ARG      COUNT1, I, 1, 300
            FUNCTION/ARG      COUNT2, I, 1, 300
            FUNCTION/ARG      COUNT3, I, 1, 300
            LOCAL/VARIABLE    TAB, I
            TAB = (COUNT1 * COUNT2) * COUNT3
            FUNCTION/END TAB
```

GOSUB (Go To Subroutine)

Syntax:

GOSUB mark

where:

argument	valid type(s)	range	description
mark	label	any valid label at the beginning of a subroutine	go to marked function with return

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

Executing a GOSUB statement saves the current position in the program execution of the user task, then begins executing subroutine statements beginning with the statement identified by the GOSUB mark.

When executing the subroutine encounters a RETURN statement, program execution resumes at the next executable program statement following the GOSUB statement that called the subroutine.

Example:

```
L00:    If I01 >= I02 gosub Sub2
        Else
           I10 = 255              ; Set I10 to 255
           gosub Sub1             ; Call subroutine one
        Endif
           .
           .
           .
Sub1:
        I01 = I01*I02             ; Multiply
        RETURN
Sub2:
        I01 = I01/I03             ; Divide
        RETURN
```

GOTO (Go To Mark)

Syntax:

GOTO <name>

where:

<name> a valid Mark identifying a unique position in a program.

Description:

The CLC stops executing the current series of program statements and immediately begins executing the series that starts with the statement identified by the mark in the GOTO instruction.

This is an unconditional branch. Since a GOTO does not save the current position in the program execution flow, branching to a subroutine or event function results in an error when the execution flow encounters the RETURN statement.

Example:

```
L00:  If I01 >= I02
         GoTo L10
      Else
         I10 = 255
         GoTo  L15
      Endif
L10:  I01 = I01*I02
      .
      .                    ;program statements
      .
      GoTo  Start1
L15: I01 = I02/I03
      .
      .                    ;alternative program statements
      .
      GoTo  Start2
```

IF (If-Else-Endif Conditional Branch)

Syntax:

IF value1, test, value2

or

PLC/TEST (reg, bit)

CAM/STATUS (cam_number, test, condition)

<program statements to execute if the expression evaluates to non-zero>

ELSE

<program statements to execute if the expression evaluates to zero>

ENDIF

where:

argument	valid type(s)	range	description
value1	integer or floating point - constant - variable Ix,I[x] or Fx,F[x] - global variable GIx,GI[x] or GFx,GF[x] - label	any valid constant, variable, or table entry	
value2	same as above		
test	ASCII characters	> >= < <= != == **(**CAM/STATUS limited to !=,==**)**	greater than greater than, or equal to less than less than, or equal to not equal equivalent to
reg	integer, constant or label, Ix, GIx	1-1024 CLC-V 1-512 CLC-P, CLC-D	Source regester to test
bit	integer, constant or label, Ix, GIx	1-16	Bit with register to test
condition	constant or label	0 - no valid cam is stored 1 - cam is being calculated 2 - cam is being sent to drive (drive cams only) 3 - cam is ready for activation 4 - cam is active and running	CAM Condition
cam_number	integer or constant	1-8	Cam to be checked

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

The IF-ELSE-ENDIF structure provides conditional execution of the program statements between the IF and ENDIF keywords, depending upon the evaluation of a test relationship. The control structure also provides an optional ELSE keyword for conditional program branching between two alternative series of program statements.

If the expression is true (has a non-zero value), and there is no ELSE keyword, the program statements between the IF and ENDIF keywords are executed. If the optional ELSE keyword is used, the statements between the IF and ELSE keywords are executed. Program execution then continues with the first program statement after the ENDIF keyword.

If the expression is false (has a zero value) and there is no ELSE keyword, the program continues with the first instruction after the ENDIF keyword. If the optional ELSE keyword is used, the program statements between ELSE and ENDIF keyword are executed. The program execution then continues with the first program statement after the ENDIF keyword.

“IF” structures may be nested up to eleven deep. All IF statements must be balanced with a matching ENDIF.

Example:

```
      .
      .
L00:     If I01 >= I02
            Goto L10
         Else
            I10 = 255
            gosub Sub2
         Endif
L10:     I01 = I01*I02
            .
            .                 ;program statements
            .
Sub2:    I01 =  I02*I03
         RETURN
```

PLC/TEST Variations:

Directly ties in the PLC/TEST commmand to an IF-Else-Endif conditional branch. Tests a bit in the specified I/O regester table enter. If PLC/TEST checks for an "on" ? condition. If !PLC/TEST checks for an "off" ? condition.

Example:

```
; PLC/TEST variations
      if( PLC/TEST 100, stewart )
         F30 = 5
      endif
      if( !PLC/TEST 100, 2 )
         F30 = 5
      endif
      if PLC/Test 100, 1
         F30 = 5
      endif
      if !PLC/Test 100, 2
         F30 = 5
      endif
      if( PLC/Test( 100, 1 ))
         F30 = 5
      endif
      if plc/test( I5, GI6 )
        else
           F31 =6
      endif
        F31= 3.1415927
```

CAM/Status Variations:

Directly ties in the CAM/STATUS command to an IF-Else-Endif conditional branch.

After a cam download or after the CAM/BUILD command, it takes some time to calculate and store the new cam. When a cam is activated on the CLC, it does not run until the cam is at the end of its cycle.

IF CAM/Status checks the status of a specified cam. The test is limited to == (equivalent to) or != (not equal) arguments.

The status integer returns the specified cam condition:

Errors will be issued when:

The cam number is not valid (out of range or drive is not configured).

Example:

```
        .
        .
        .
  ; CAM/Status variations
        if cam/status 1 != 4
           F30 = 5
        endif
        if cam/status ( 1) == 4
           F30 = 5
        endif
        if (CAM/STATUS ( 1) == 4 )
           F30 = 5
        endif
```

KINEMATIC (Use a Kinematic Definition for a Task)

Syntax:

KINEMATIC kinematic

where:

argument	valid type(s)	range	description
kinematic	integer - constant - label	a valid kinematic library number	

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

Each task can have its own kinematic to allow for motion in Cartesian space. This instruction tells the path planner which set of equations in the optional kinematic library to use for motion.

This instruction is active at the start of a Task, and is removed before normal system cycling.

The KINEMATIC statement is allowed only in the main Tasks: A, B, C, or D.

Example:

```
        .
        .
        .
  kinematic    10     ; Use kinematic library number 10
        .
        .
        .
```

LOCAL/VARIABLE

Syntax:

LOCAL/VARIABLE label, type

where:

argument	valid type(s)	range	description
label	ASCII string	1 to 20 characters	name of variable used as local variable
type	ASCII character	'I' = integer 'F' = floating point 'ABS'= absolute point index 'REL'= relative point index	type of variable used as local variable

Description:

Declares a local variable including its name and type. A total of sixteen function arguments and local variables may be used per function.

Example:

```
;  Main task calling subroutine and returning value

Task_A:     TASK/START A
            I1 = CALL         sub, 5, 10, 20
            TASK/END A

;  Subroutine to multiply input values

sub:  FUNCTION/START U
            FUNCTION/ARG      COUNT1, I, 1, 300
            FUNCTION/ARG      COUNT2, I, 1, 300
            FUNCTION/ARG      COUNT3, I, 1, 300
            LOCAL/VARIABLE    TAB, I
            TAB = (COUNT1 * COUNT2) * COUNT3
            FUNCTION/END TAB
```

MESSAGE/DIAG (Task Diagnostic Message Definition)

Syntax:

MESSAGE/DIAG message

where:

argument	valid type(s)	range	description
message	ASCII	up to 79 characters	diagnostic text message

* **VM 5.0** {MESSAGE type,"text", [variable]}.

Types are: 1=status, 2=diagnostic

Text must be in quotes. Messages may have one formatted variable in its string. Use "%s" to format optional variable. Previous syntax supported for upward compatibility.

Description:

The MESSAGE/DIAG directive provides a user program with the ability to send a diagnostic message to the user that is associated with a specific statement in the user program. MESSAGE/DIAG embeds a tag or index into the instruction following the MESSAGE/DIAG statement.

A diagnostic message provided by the CLC can be used by the user interface to select and display the message for system maintenance or troubleshooting.

The diagnostic message may also be used for program debugging during program development in a manner similar to embedding print statements at critical points in the program.

Example:

```
      .
      .
Move/line      ABS[1]
Message 2, "Waiting for safe zone bit in PLC to be true"
;
;The diagnostic message lets the user know that the program has advanced
;to the program statements that check the safe zone I/O bit.
;
L10:    Plc/read       I01, 0
        If I01 & 0x100
           Goto L20
           Delay          1000
           Goto           L10
        EndIf
L20:
      .
```

MESSAGE/STATUS (Task Status Message Definition)

Syntax:

MESSAGE/STATUS message

where:

argument	valid type(s)	range	description
message	ASCII	up to 79 characters	status text message

* **VM 5.0** {MESSAGE type,"text", [variable]}.

Types are: 1=status, 2=diagnostic

Text must be in quotes. Messages may have one formatted variable in its string. Use "%s" to format optional variable. Previous syntax supported for upward compatibility.

Description:

The MESSAGE/STATUS directive is similar to the MESSAGE/DIAG directive. MESSAGE/STATUS also embeds a tag or index into the instruction following the MESSAGE/STATUS statement. However, MESSAGE/STATUS character strings are stored in a different table than MESSAGE/DIAG messages and provide a different series of message index numbers.

By providing two tables, status messages may be used for prompting the system operator, diagnostic messages for program debugging and system maintenance.

Example:

```
        .
        .
        .
      Move/line      ABS[1]
      Message 1, "Waiting for safe zone bit in PLC to be true"

  ;The status message lets the user know that the program has advanced
  ;to the program statements that check the safe zone I/O bit.

  L10: Plc/read       I01, 0
       If I01 & 0x100
          Goto L20
          Delay          1000
          Goto           L10
       EndIf
  L20:
        .
        .
        .
```

MOVE/CIRCLE (Coordinated Move with Circular Interpolation)

Syntax:

MOVE/CIRCLE ABS[index1], ABS[index2]

or

MOVE/CIRCLE REL[index1], REL[index2]

where:

argument	valid type(s)	range	description
index1	integer - constant - variable Ix - global variable GIx - label	a valid absolute or relative point table entry	defines the mid-point of the circular arc
index2	integer - constant - variable Ix - global variable GIx - label	a valid absolute or relative point table entry	defines the end-point of the circular arc

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

The MOVE/CIRCLE instruction provides circular motion along a path in Cartesian space defined by three sets of coordinates. Program modifiable integer variables may be used to specify the starting and ending coordinates in the point tables.

The first form provides circular movement using absolute coordinates. Motion occurs from the end point of the last move or current position, through the absolute coordinate specified by the table reference of the first argument, and ends at the absolute coordinate specified by the table reference of the second argument.

The second form allows motion to begin at a relative offset from the end of the last move or current position, moves through a relative offset specified by the table reference of the second argument, and ends at an absolute coordinate specified by the table reference of the third argument.

Example:

```
      .     .
;enter points into absolute point table
ABS[0]x     =     0.0
ABS[1]y     =     1.0
ABS[1]z     =     0.0
      .
      .     .;assume the current position is  -1.0, 0.0, 0.0
MOVE/CIRCLE ABS[1], ABS[2]
;the move results in a semi-circular move
;from x=-1, y=0, through x=0, y=1; to x=1, y=0
```

MOVE/JOINT (Coordinated Move Joint Point to Point)

Syntax:

MOVE/JOINT ABS[index]

where:

argument	valid type(s)	range	description
index	integer - constant - variable Ix - global variable GIx - label	a valid absolute point table entry	specifies the endpoint of move

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

MOVE/JOINT is an absolute point-to-point move, with only the endpoint of the move specified. The actual path taken to the specified point is indeterminent (i.e., not linear or circular) and may assume whatever form the path planner requires; however, once programmed and planned, the path is repeatable. The path is optimized to minimize time and uses accel and slew rates for the coordinated axes (as opposed to the line and circle coordinated motion commands which use the world rates).

Most commonly used with robotic applications, the move Joint command is also the only method of elbow repositioning.

Example:

```
        .
        .
        .
ABS[1].X    =     10.0        ;\
ABS[1].Y    =     20.0        ; -> Sets Absolute point 1
ABS[1].Z    =     15.0        ;/
        .
        .
        .
MOVE/JOINT  ABS[1]            ; An absolute point-to-point
                  ; Coordinated motion
        .
```

MOVE/LINE (Coordinated Move with Straight Line Interpolation)

Syntax:

MOVE/LINE ABS[index]

or,

MOVE/LINE REL[index]

where:

argument	valid type(s)	range	description
index	integer - constant - variable Ix - global variable GIx - label	a valid absolute or relative point table entry	specifies the endpoint of move

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

The MOVE/LINE instruction provides motion along a straight line path in Cartesian space, defined by two sets of coordinates.

The first form provides linear movement to absolute coordinates. Motion occurs from the end point of the last move or current position, and ends at the absolute coordinate specified by the table reference of the last argument.

The second form begins motion at a relative offset from the end of the last move or current position, and ends at an absolute coordinate specified by the table reference of the last argument.

Note that program modifiable integer variables may be used to specify the starting and ending coordinates in the point tables.

Example:

```
        .
        .
        .
  ;assume that the current position is·-1.0, 0.0, 0.0

  Move/Line   ABS[1]
  ;This results in a linear move from x=-1, y=0; to x=0, y=1
        .
        .
        .
```

PARAMETER/BIT (Initialize Parameter Bit)

Syntax:

PARAMETER/BIT type, set, param, source, bit

where:

argument	valid type(s)	range	description
type	ASCII character	A C D T	Parameter Type - A for Axis parameters - C for System parameters - D for Drive parameters - T for Task parameters
set	integer,or label	1 Ix,I[x],GIx, GI[x] A, B, C or D	Parameter Set 1 for System parameters a valid drive or axis number for Drive or Axis parameters A, B, C, or D character for Task parameters
param	integer	a valid parameter	parameter ID number
source	integer - constant - variable Ix,I[x],GIx,GI[x] - global variable Ix,I[x],GIx,GI[x] - label	zero or, non-zero	logical value for bit - zero value clears the bit - non-zero sets the bit
bit	HEX format or, label	decimal 1 - 16, or hexadecimal, (0X0001 to 0xFFFF)	decimal selects a single bit hexadecimal can select multiple bits e.g., 0x0300 = bits 9 and 10 0x2003 = bits 1,2, and 14

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

PARAMETER/BIT is used to set one or more bits in a parameter when the program is activated. The instruction tests for a zero/non-zero logical constant or variable to set or clear the bit(s) enabled by a specified bit mask.

Example:

```
TASK_A: TASK/START A
PLC/CLEAR   120, 8            ; Clears bit 8 of register 120
      .
      .
PARAMETER/BIT     A, 1, 4, I1, 0x0002
                              ; Sets Rotary mode for axis 1
      .
      .
AXIS/RATIO  1, 2, 2, 1        ; Axis 2 is linked to axis 1 at 1/2
      .
      .
      .
TASK/END    A
```

PARAMETER/GET (Load Parameter to a Variable)

Syntax:

PARAMETER/GET type, set, param, target

where:

argument	valid type(s)	range	description
type	ASCII character	A C D T	Type of parameter - A for Axis parameters - C for System parameters - D for Drive parameters - T for Task parameters
set	integer, or label	1 Ix,I[x],GIx,GI[x] A, B, C or D	Parameter set - 1 for System parameters - a valid drive or axis number for Drive or Axis parameters - A, B, C, or D character for Task parameters
param	integer	a valid parameter	parameter ID number
target	integer or floating point - variable Fx,F[x], Ix, I[x] - global variable Gfx, GF[x] - label	a valid variable	target variable to load with parameter value target variable data type must match parameter data type

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

Retrieves a parameter from the CLC's system, task or drive parameter tables and stores the value in the specified variable.

Incorrectly specifying the parameter type will result in a compiler error.

Example:

```
        .
        .
        .
      Parameter/Get      D, 1, STATUS, I01
      If I01 == 0
         Goto L10
      Else
         Gosub new_sub
      Endif
        .
        .
        .
L10:  Move/Line ABS[2]
      I01 = 100
      Parameter/Set      D, 1, POSITION, I01
        .
        .
        .
```

PARAMETER/INIT (Initialize a Parameter)

Syntax:

PARAMETER/INIT type, set, param, source

where:

argument	valid type(s)	range	description
type	ASCII character	A C D T	Type of parameter - A for Axis parameters - C for System parameters - D for Drive parameters - T for Task parameters
set	integer, or label	1 Ix,I[x],GIx,GI[x] A, B, C or D	Parameter set - 1 for System parameters - a valid drive or axis number for Drive or Axis parameters - A, B, C, or D character for Task parameters
param	integer	a valid parameter	parameter ID number
source	integer or floating point - constant - variable Fx,F[x] - global variable GFx,GF[x],GIx,GI[x] - label	a valid variable	source variable to load to parameter source variable data type must match parameter data type

Description:

PARAMETER/INIT is used to load the source value into the specified parameter when the program is activated.

Example:

```
        .
        .
PARAMETER/INIT    A, 1, 120, 32.0
                                ; Sets axis 1 parameter 120 to
                          ; initialize acceleration
```

PARAMETER/SET (Set a Parameter)

Syntax:

PARAMETER/SET type, set, param, source

where:

argument	valid type(s)	range	description
type	ASCII character	 A C D T	Type of parameter - A for Axis parameters - C for System parameters - D for Drive parameters - T for Task parameters
set	integer, or label	 1 integer, or label A, B, C or D	Parameter set - 1 for System parameters - a valid drive or axis number for Drive or Axis parameters - A, B, C, or D character for Task parameters
param	integer	a valid parameter	parameter ID number
source	integer or floating point - constant - variable Fx,F[x] - global variable GFx,GF[x],GIx,GI[x] - label	a valid variable	source variable to load to parameter source variable data type must match parameter data type

Description:

PARAMETER/SET updates the CLC's system, task or drive parameter table with a new value for the specified parameter.

Incorrectly specifying the parameter type will result in a compiler error.

Example:

```
        .
        .
        .
Parameter/Get D, 1, STATUS, I01
If I01 == 0 Goto L10
Else
      Gosub New_Subroutine
        .
        .
        .
L10: Move/Line ABS[2]
I01 = 100
Parameter/Set d, 1, POSITION, I01
        .
        .
        .
```

PATH/ABORT (Aborts Coordinated Motion)

Syntax:

PATH/ABORT task

where:

argument	valid type(s)	range	description
task	ASCII character	A, B, C or D	task ID character

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

PATH/ABORT decelerates motion on path then aborts all segments placed in the task's queue by the path planner. Motion must be restarted with a cycle start, it cannot be continued. Any events are lost.

Example:

```
TASK_D:TASK/START D
       .
       .
REL[1].X    =     1.0         ; Sets relative point 1
       .
       .
IF (I1 == 3)
      PATH/ABORT  D           ; Aborts task D operation
```

PATH/POSITION (Get Current Path Absolute Position)

Syntax:

PATH/POSITION task, ABS[index]

where:

argument	valid type(s)	range	description
task	ASCII character	A, B, C or D	task ID
index	integer - constant - variable Ix - global variable GIx - label	a valid absolute point table entry	index to a target absolute position table entry

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

The current position of the path planner is returned to the specified point table entry. The current contents of the point table are overwritten. The other current specifications in the point table (rate. accel/decel, events, etc.) are not affected.

One task may obtain the position of another task by specifying the desired task ID.

Example:

```
L10:  path/position 1,ABS[6]              ; Get current position
      F[01] = ABS[6].x*ABS[6].x
      F[01] = F[01] + ABS[6].y*ABS[6].y
      F[01] = F[01] + ABS[6].z*ABS[6].z
      F[01] = SQRT( F[01] );              ; Compute vector length
      if F[01] >= 100.0 the
      goto L10  ; Loop if larger than 100
```

NOTE: *Add one more point to this instruction.*

Example:

```
ABS[x] defined in Path/Position instruction =
# of absolute points defined in Data/Size +1
```

PATH/RESUME (Resume Coordinated Motion)

Syntax:

PATH/RESUME task

where:

argument	valid type(s)	range	description
task	ASCII character	A, B, C or D	task ID character

Description:

PATH/RESUME continues coordinated motion halted by a PATH/STOP command. Any queued time or distance-based events saved by the PATH/STOP will also resume when motion continues.

Example:

```
TASK_C:TASK/START C
      .
      .
PATH/STOP   A                ; Stops Coordinated motion of task A
DELAY       3000             ; Waits for 3 seconds
PATH/RESUME A                ; Resumes Coodinated motion of task A
      .
      .
      .
```

PATH/STOP (Halt Coordinated Motion)

Syntax:

PATH/STOP task

where:

argument	valid type(s)	range	description
task	ASCII character	A, B, C or D	task ID character

Description:

PATH/STOP commands the path planner to decelerate motion and halt motion on the path. PATH/STOP saves the "look-ahead" path planned by the path planner and any associated time or distance-based events. Motion and events on the path may be resumed by a PATH/RESUME command.

Example:

```
SAMPLE_RATE EQU   0.0          ; Sets equate
      .
      .
TASK_A:TASK/START A
      .
      .
IF (SAMPLE_RATE >= 1000.0)
      PATH/STOP   B            ; Halts Coordinated motion of task B
            .
            .
TASK/END    A
```

PATH/WAIT (Pause Program for Motion)

Syntax:

PATH/WAIT task, ABS[index], state

or

PATH/WAIT task, REL[index], state

where:

argument	valid type(s)	range	description
task	ASCII character	A, B, C or D	task ID
index	integer - constant - variable Ix - global variable GIx - label	a valid absolute or relative point table entry	point table entry specifying position to pause program
state	integer -label	0 to 8	requested state of path planner 0 = segment ready 1 = acceleration 2 = slew (constant speed) 3 = blending 4 = target deceleration 5 = controlled stop 6 = stopped 7 = at target 8 = done

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

PATH/WAIT tests the current state of the path planner for a specified point. Since the path planner is typically one or more segments ahead of physical motion, PATH/WAIT can be used to temporarily halt program execution until the path planner begins a specific type of processing for a specified point.

Example:

```
TASK_D:TASK/START D
      .
      .
MOVE/LINE   ABS[1]            ; Straight Line Coordinated motion
PATH/WAIT   D, ABS[1], 8      ; Waits until point is done
      .
      .
      .
TASK/END    D
```

PID/CONFIGURE

PID/CONFIGURE #, type, control_register, status_register, loop_time, set_point_type, set_point, set_point_axis, feedback_type, feedback, feedback_axis, output_type, output, output_axis, control_block

Example:

```
PID/CONFIGURE 1,1.130, 131, 8, 1, F5, 0, 1, F6, 0, 1, F7, 0, F10
```

See also Section
VAR/INIT

PLC/CLEAR (Clear I/O Register Bit)

Syntax:

PLC/CLEAR register, bit

where:

argument	valid type(s)	range	description
register	integer - constant - label	a valid I/O register	specifies an I/O register
bit	integer - constant - label	1 to 16	specifies a bit in the register

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

PLC/CLEAR clears the specified bit in the specified I/O register table entry to 0.

Example:

```
        .
        .
        .
Cycle_On    equ   10          ; Cycle on register
Go          equ   6           ; Bit for go
        .
        .
        .
Plc/Clear   Cycle_On, Go      ; Clear bit 6 of cycle on
        .
        .
        .
```

PLC/READ (Read I/O Register(s))

Syntax:

PLC/READ register, count, target

where:

argument	valid type(s)	range	description
register	integer - constant - label	a valid I/O register	first register to read
count	integer - constant - label	1 to maximum number of I/O registers	number of registers to read
target	integer - variable Ix, GIx, I[x], GI[x] - label	a valid integer variable	first target variable

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

PLC/READ copies the contents of one or more sequential 16-bit registers in the CLC I/O registers table to sequential entries in the integer variable table.

Example:

```
        .
        .
        .
iobase      equ   100
count       equ   20
define      I5,ioregs
;
plc/read    iobase, count, ioregs   ;Read the 20 CLC I/O registers, from
                                    ;100 through 120 to integer variables
                                    ;I05 through I25
        .
        .
        .
```

This statement copies the contents of twenty 16 bit PLC I/O table entries (I/O registers 101 through and including 120) to the integer variables I01 through and including I20.

PLC/SET (Set I/O Register Bit)

Syntax:

PLC/SET register, bit

where:

argument	valid type(s)	range	description
register	integer - constant - label	a valid I/O register	specifies an I/O register
bit	integer - constant - label	1 to 16	specifies a bit in the register

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

PLC/SET sets a specified bit in the specified CLC I/O register table to 1.

Example:

```
Light_Reg  equ 20
Light_On   equ  5
      .
      .
      .
plc/set    Light_Reg, Light_On     ; Turn on bit 5 of register 20
      .
      .
      .
```

PLC/TEST (Test I/O Register Bit)

Syntax:

PLC/TEST register, bit, target

where:

argument	valid type(s)	range	description
register	integer - constant - label - variable Ix, GIx	a valid I/O register	source register to test
bit	integer - constant - label - variable Ix, GIx	1 to 16	bit within register to test
target	integer - variable Ix, I[x], GIx,GI[x] - label	a valid integer variable	variable to receive binary (1 or 0) test result

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

PLC/TEST tests a bit in the specified I/O register table entry and returns a logical value to the specified integer variable according to the state of the I/O register bit. (The value of the integer variable may then be used to control program flow.)

Example:

```
        .
        .
        .
Plc/Test    20, 5, I03        ;set I03 flag to I/O register, bit 5 state
If I03 > 0 Gosub calculate    ;do calculate if flag is set
Else
   gosub subtract
EndIf
        .
        .
        .
```

PLC/WAIT (Pause Program for I/O)

Syntax:

PLC/WAIT register, bit, state

where:

argument	valid type(s)	range	description
register	integer - constant - label	a valid I/O register	source register to test
bit	integer - constant - label	1 to 16	bit within register to test
state	integer - variable Ix, I[x], GIx,GI[x]	zero or one	variable containing the logical value required to resume program execution

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

PLC/WAIT may be used to pause program execution with a task until the state of the specified I/O bit equals the specified logical value.

PLC/WRITE (Write to I/O Register(s)

Syntax:

PLC/WRITE register, count, source

where:

argument	valid type(s)	range	description
register	integer - constant - label	a valid I/O register	starting target register to write
count	integer - constant - label	1 to maximum number of registers	number of registers to write
source	integer - constant - variable Ix, I[x], GIx,GI[x] - label	a valid integer variable	starting source variable

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

PLC/WRITE copies the contents of one or more sequential integer variables from the integer variable table to the I/O register table.

Example:

```
I[01]  =  0x0001
I[02]  =  0x6666
      .
      .
      .
Plc/Write   2, 3,I[01]
      .
      .
      .
```

This statement writes integer variable I[01] to I/O register 3, and I[02] to I/O register 4.

PLS/INIT (Compiler directive)

Syntax:

PLS/INIT (PRIMARY) switch, 0, register, type, axis, offset

or

switch, element, on position, off position (SECONDARY)

where:

argument	valid type(s)	range	description
switch	integer - constant - label	1	
register	integer - constant - label		output register
type *	integer - constant - variable Ix,	1 - ELS 2 - virtual axis 3 - primary axis 4 - secondary axis	type of axis that the limit switch is tracking
axis		0 if type 1 or 2 else 1-40	axis number
offset		0-360	PLS phase advance

or

argument	valid type(s)	range	description
switch	integer - constant - label	1	
element	integer - constant - label	1-16	switch element outputs (output register bits)
on position	integer - constant - variable Ix,	0-360	position in which switch element turns on
off position	integer - constant - variable Ix	0-360	position in which switch element turns off

*** Currently ELS is the only valid 'type'.**

Description:

Both the primary and secondary PLS setup commands are executed by Visual Motions compiler to initialize the PLS table. The location of the PLS/INIT command is not important because only the Visual Motion compiler uses it. Use the runtime commands (PLS[1].t=1) to change the table in the program flow. The PLS is disabled while the output register is zero or when all bits have zero in both the on and off positions.

A programmable limit switch supports 16 outputs and a phase advance. The position input for the PLS is the ELS master, which can be an external encoder, an axis feedback or a virtual master. The outputs are updated every SERCOS cycle.

Example:

```
Task_A:TASK/START A
      local/variable    var1, I
      data/size 50, 50, 20, 20, 10, 0
;Primary PLS setup command
      pls/init 1,0,120,1,0,0.0
;Secondary PLS setup command
      pls/init 1,1,20,40
      pls/init 1,2,20,40
      pls/init 1,3,20,40
      pls/init 1,4,20,40
      pls/init 1,5,20,40
      pls/init 1,6,20,40
      pls/init 1,7,20,40
      pls/init 1,8,20,40
      pls/init 1,9,20,40
      pls/init 1,10,20,40
      pls/init 1,11,20,40
      pls/init 1,12,20,40
      pls/init 1,13,20,40
      pls/init 1,14,20,40
      pls/init 1,15,20,40
      pls/init 1,16,20,40
;Runtime commands
      PLS[1].r=120                        ;register
      PLS[1].o=30.0                       ;offset
      PLS[1].on1=30
      PLS[1].on1=30
      PLS[1].on16=120
      PLS[1].off1=120
      PLS[1].off16=300
            var1 =call        loop25
      TASK/END A
.
.
```

REGISTRATION

Syntax:

REGISTRATION axis, controlRegister, statusRegister, floatBlock, IntBlock

Description:

See Registration Icon.

See also VAR/INIT.

```
VAR/INIT    floatBlock,     ; Sets up float variables at compile time
            probePosition
            probeWindowStart
            probeWindowStop
            correctionWindowStart
            correctionWindowStop
            maximumCorrection
```

Default label naming scheme and comment for Registration

15 float variables

```
RGIS_01_ PROSITION      ;Registration, Axis 01, expected mark position
RGIS_01_ WIN_START      ;Registration, Axis 01, distance before expected to look
RGIS_01_ WIN_STOP       ;Registration, Axis 01, distance after expected to look
RGIS_01_ COR_START      ;Registration, Axis 01, correction to start at this point
RGIS_01_ COR_STOP       ;Registration, Axis 01, correction to be completed before
RGIS_01_ MAX_CORRECT    ;Registration, Axis 01, maximum correction per peroid
RGIS_01_ SETPOINT       ;Registration, Axis 01, calculated expected position
RGIS_01_ COR_ERROR      ;Registration, Axis 01, calculated  correction distance
RGIS_01_ PROBE_VALUE    ;Registration, Axis 01, feedback position at probe 1 trigger
RGIS_01_ RESERVE_F6     ;Registration, Axis 01, reserve float 6
RGIS_01_ RESERVE_F5     ;Registration, Axis 01, reserve float 5
RGIS_01_ RESERVE_F4     ;Registration, Axis 01, reserve float 4
RGIS_01_ RESERVE_F3     ;Registration, Axis 01, reserve float 3
RGIS_01_ RESERVE_F2     ;Registration, Axis 01, reserve float 2
RGIS_01_ RESERVE_F1     ;Registration, Axis 01, reserve float 1
```

4 integer variables

```
RGIS_01_ FLAG1          ;Registration, Axis 01, flag word 1
RGIS_01_ MISS_MARKS     ;Registration, Axis 01, number of missing marks allowed
RGIS_01_ COR_TYPE       ;Registration, Axis 01, correction type
RGIS_01_ RESERVE_I1     ;Registration, Axis 01, reserve integer 1
```

RETURN (Return From Subroutine)

Syntax:

RETURN

Description:

A RETURN statement must be entered as the last instruction of a subroutine. When the subroutine execution encounters a RETURN statement, program flow exits the subroutine and resumes with the program statement following the GOSUB statement that called the subroutine.

Example:

```
        .
        .
        .
Sub1: Move/Line   ABS[2]
            Plc/Set          1,1
            Return
        .
        .
        .
```

ROBOT/ORIGIN

Syntax:

ROBOT/ORIGIN REL[index]

where:

argument	valid type(s)	range	description
index	integer - constant - variable Ix - global variable GIx - label	a valid relative point table entry	defines the zero frame of reference for a robot

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

The ROBOT/ORIGIN instruction is used in coordinated motion programs to construct a zero frame of reference from the x, y, z, roll, pitch and yaw coordinates of a relative point. This moves the effective origin of the robot from the default to the location specified by the programmed relative point.

For example, if REL[3] = {1, 2, 3, 0, 0, 0, ...} and this point is specified as the robot origin, then the robot origin would be offset by one unit along the x axis, two units along the y axis and three units along the z axis. Once the instruction is executed, all jogging, teaching and path locations are affected by the new origin.

ROBOT/TOOL

Syntax:

ROBOT/TOOL REL[index]

where:

argument	valid type(s)	range	description
index	integer - constant - variable Ix - global variable GIx - label	a valid relative point table entry	defines the tool frame of reference for a robot

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

The ROBOT/TOOL instruction is used in coordinated motion programs to construct a tool frame of reference from the x, y, z, roll, pitch and yaw coordinates of a relative point. This moves the effective end-of-arm tool to the location specified by the programmed relative point.

For example, if REL[4] = {0, 0, 10, 0, 0, 0, ...} and this point is specified as the robot tool location, then the robot tool location would be offset by ten units along the z axis from the faceplate of the robot. Once the instruction is executed, all jogging, teaching and path locations are affected by the end-of-arm tool location.

SEQUENCER

Syntax:

SEQUENCER seq_name

where:

argument	valid type(s)	range	description
seq_Name	integer - constant - variable Ix - global variable GIx - label	determined by program limits	sequencer name

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

The sequencer name is a number or label equating to a number. The number has a range of 1 to n, where n is the number of sequencers defined in the Size icon.

Example:

```
;     .
;     .
; Run sequencer IVORY
      SEQUENCER   IVORY
      TASK/END    A
```

SEQ/LIST

Syntax:

SEQ/LIST seq_name, list_number, list_name

or

seq_name, 0, count

where:

argument	valid type(s)	range	description
seq_Name	integer - constant - variable Ix - global variable GIx - label	determined by program limits	sequencer name
list_Number	integer	valid list number or 0 for count	number of step list
list_Name	label or count (integer)		name of step list or if range=0 this is count (total number of lists)

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

This instruction assigns a sequencer step list to an existing sequencer along with a number which determines the order the sequencer will follow when executed. The name of the step list is also included in this command.

A step list can be used by more than one sequencer. It can also be used repeatedly within a given sequencer.

Example:

```
IVORY equ   1
Task_A:     TASK/START  A

;Declare 10 sequencers,10 steps, 50 function steps

      DATA/SIZE   50,50,0,0,4,0,10,10,50

; Build sequencer IVORY

; Sequencer IVORY will have 5 steps

      SEQ/LIST    IVORY,0,5
      SEQ/LIST    IVORY,1,Mold_Open
      SEQ/LIST    IVORY,2,Move_In
      SEQ/LIST    IVORY,3,Grab_Part
      SEQ/LIST    IVORY,4,Move_to_Drop
      SEQ/LIST    IVORY,5,Drop_Part
....
```

SEQ/STEP

Syntax:

SEQ/STEP list_name, step_number, function_name, arg1, arg2, ...arg5

or

list_name, 0, count

where:

argument	valid type(s)	range	description
list_Name	label		name of step list
step_Number	integer or 0		
function_Name or count	label or count (integer)	any valid function	name of function or if step_number=0 this is count (the total number of steps)
arg1, arg2, ...arg5	integer, float - constant	determined by program limits	function arguments

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

This command defines a sequencer step within a step list. It identifies an existing function or subroutine and passes on up to five function arguments when executed. If the given step_number is

Example:

```
.
; Step Mold_Open will have 4 functions in it

       SEQ/STEP     Mold_Open,0,4
       SEQ/STEP     Mold_Open,1,vac_on,1, , , ,
       SEQ/STEP     Mold_Open,2,waitcool,1000, , , ,
       SEQ/STEP     Mold_Open,3,chk_part, , , , ,
       SEQ/STEP     Mold_Open,4,move_rdy, , , , ,

; Step Move_In will have 4 functions in it

       SEQ/STEP     Move_In,0,4
       SEQ/STEP     Move_In,1,chk_mold, , , , ,
       SEQ/STEP     Move_In,2,movechk,1,1000, , ,
       SEQ/STEP     Move_In,3,inmoldms,100,555, , ,
       SEQ/STEP     Move_In,4,vac_on,4, , , ,

; Step Drop_Part will have 1 function in it

       SEQ/STEP     Drop_Part,0,1
       SEQ/STEP     Drop_Part,1,f1234567890123456789, , , , ,
;      .
;      .
```

TASK/AXES (Task Axes Definition)

Syntax:

TASK/AXES type, axis1, {axis2, axis3}

where:

argument	valid type(s)	range	description
type	integer -label	1 to 6	type of motion 1 - single axis motion 2 - coordinated motion 3 - velocity mode (no positioning) 4 - for ratioed axes (master/slave) 5 - ELS Slave mode 6 - Torque Mode
axis1 axis2 axis3	integer - constant - label	1 to the maximum number of valid axes	

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

TASK/AXES defines the axes that are used within a task and how they are used. A maximum of six axes may be assigned using one TASK/AXES statement. A task may contain several TASK/AXES statements when the task requires additional axes or uses axes in ratio mode.

When type 4 (ratio mode) is specified only two axis arguments may be supplied; the second argument is the master axis and the third argument is the slave axis. Axes used in ratio mode must have been previously defined (in any task) by another TASK/AXIS statement. The mathematical ratio between the two axes must be set by an AXIS/RATIO statement.

Tasks that do not use motion control do not require a TASK/AXES statement. However, all drives connected to the SERCOS ring must be registered to the CLC by declaring each drive to its axis in a TASK/AXES statement, even if the axes are not used. This insures that the CLC will correctly respond to drives automatically identified by the SERCOS initialization procedures.

The TASK/AXES statement may be used only within the main Tasks: A, B, C, or D.

This instruction is only active during download and is removed from the instruction stream before normal program cycling.

Example:

```
TaskB_Ax1   equ 3                                ; Define Task B axis one
TaskB_Ax2   equ 4                                ; Define Task B axis two
TaskB_Ax3   equ 6                                ; Define Task B axis three
      .
      .
      .
Task_B:
task/start  B                                    ; Start B
task/axes   TaskB_Ax1, TaskB_Ax2, TaskB_Ax3      ; Assign axes to task B
      .
      .
      .
```

TASK/END (Mark the End of a Task)

Syntax:

TASK/END <task>

where:

argument	valid type(s)	range	description
task	ASCII character	A, B, C or D	task ID character

Description:

TASK/END indicates the end of a task, but not the end of the program. Event(s) can be outside of the instruction. All CLC instructions must appear between TASK/START and TASK/END statements that specify one of the four tasks.

Active during download, this instruction is removed from the instruction stream before normal program cycling.

Example:

```
Task_A:
Task/Start  A
      .           ;
      .           ;Task A's program statements
      .           ;
Task/End    A
;
Task_B:
Task/Start  B
      .
      .
      .
```

TASK/START (Define the Start of a Task(s))

Syntax:

TASK/START <task>

where:

argument	valid type(s)	range	description
task	ASCII character	A, B, C or D	task ID character

Description:

TASK/START indicates the start of a task, but not the end of the program. Event(s) can be outside of the instruction. All CLC instructions must appear between TASK/START and TASK/END statements that specify one of the four tasks. Each task must be identified by a "Task_(task letter):" mark at the beginning of the task's program.

Active during download, this instruction is removed from the instruction stream before normal program cycling.

Example:

```
Task_A:
Task/Start  A
      .           ;
      .           ;program statements
      .           ;
Task/End    A
;
Task_B:
Task/Start  B
      .
      .
      .
```

VAR/INIT

VAR/INIT var_start, arg1, arg2, arg3,....., arg20

var_start: First variable in a block of program variables(Fx, Ix) to be initialized.

arg1- arg20: initializing value.

See Registration Command

Example:

```
VAR/INIT F10, 1, 0, 1, 0, 1, 0, 0, 0, 0, 32000, 0, 1, 0
```

This instruction is used to initialize Ix and Fx values at compile time, it can be used to initialize the control block of the PID. For type 1 PID arg1 to arg13 have the following definitions:

arg1: Command scaler value.

arg2: Command bais value.

arg3: Feedback scaler value.

arg4: Feedback bais value.

arg5: Kp value.

arg6: Ki value.

arg7: Kd value.

arg8: Ki limit value.

arg9: Minimum output value.

arg10: Maximum output value.

arg11: Preset value.

arg12: Output scaler value.

arg13: Output bais value.

VME/BROADCAST (Broadcast Event to VME)

Syntax:

VME/BROADCAST VME_Units, priority, event

Where:

argument	valid type(s)	range	description
VME_Units	ASCII hexadecimal -label	0x0001 to 0xFFFF	selects one or more VME units to receive the broadcast event a selection mask is calculated by inclusive "ORing" each unit number shifted left two bit positions
priority	ASCII character	A, B, C or D	sets priority according to task identifier A = highest, D = lowest
event	floating point - constant - variable Ix,I[x] - global variable GIx,GI[x] - label	1 to 63	an event index to one of the first 63 events

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

VME/BROADCAST is used to send a broadcast event trigger that can be received simultaneously, by one or more CLC cards on a VME bus. The event functions must be enabled on the target CLC cards.

Example:

```
DEFINE      I1, EVT1          ; Defines label 'EVT1'

TASK_C:TASK/START C
DATA/SIZE   2, 2, 5, 3, 3, 0  ; Allocates memory for task C
      .
      .
VME/BROADCAST     0, C, EVT1  ; CLC unit 0 is affected
      .
```

VME/EVENT (Send Event to VME Unit)

Syntax:

VME/EVENT VME_Unit

Where:

argument	valid type(s)	range	description
VME_Unit	integer - constant - variable - global variable - label	0 to 15	specifies the target unit on the VME bus

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

VME/EVENT sends an event trigger to the specified VME unit number using CLC mailbox 4 in the VME short address space. This event must be present and enabled on the target card. The event is bound to the last task enabling the event.

Example:

```
TASK_B:TASK/START B
      .
      .
IF(TEST == SUCCESS)
      VME/EVENT   1          ; CLC unit 1 is affected
      .
      .
TASK/END    B
```

VME/READ (VME Bus Data Read)

Syntax:

VME/READ Axx, address, Dxx, format, byte_order, count, target

Where:

argument	valid type(s)	range	description
Axx	ASCII characters	A32, A24, or A16	VME source address width
address	hexadecimal Ix,I[x],GIx,GI[x]	depends on address width	VME source memory address
Dxx	ASCII characters	D32, D16, or D8	VME source data width
format	ASCII characters	F32, I32, I16, I8, U32, U16, U8, or Point	VME source data format
byte_order	ASCII character	M or I	M = Motorola (big endian) I = Intel (little endian)
count	integer - constant - variable Ix, I[x],GIx, GI[x] - label		number of data items to read
target	- variable Ix, I[x] or Fx,F[x] - global variable GIx,GI[x] or GFx,GF[x] - absolute point table ABS[n] - relative point table REL[n] - register Rx, R[x] - label		the starting address of the first item in the list of data to read

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

VME/READ copies data, such as one or more of the integer or floating point variables, or absolute or relative point table entries, from the specified VME memory address to the specified target CLC local memory.

The instruction specifies the number and width of the data items transferred. Proper word and double word byte order for 16 and 32 bit data sizes is controlled by the M or I (big or little endian) argument. Therefore, both Motorola and Intel architectures are easily supported.

For more information on the limitations of direct VME memory accesses see the CLC/VME Direct Data Access section in the chapter on CLC/VME SYSTEMS.

Format specifiers:

F32	32-bit	floating	point
I32	32-bit		integer
I16	16-bit		integer
I8	8-bit		integer
U32	32-bit	unsigned	integer
U16	16-bit	unsigned	integer
U8	8-bit unsigned integer		

Example:

```
VME/READ    A32,0x402000, D16, U16, M, 10, I01
```

Ten sixteen bit unsigned data items are read as integers into variables I[01] to I[11]. (Since the CLC uses a 680x0 "M" is specified and byte swapping is not required.)

VME/WRITE (VME Bus Data Write)

Syntax:

VME/WRITE Axx, address, Dxx, format, byte_order, count, source

Where:

argument	valid type(s)	range	description
Axx	ASCII characters	A32, A24, or A16	VME source address width
address	hexadecimal Ix,I[x],GIx,GI[x]	depends on address width	VME source memory address
Dxx	ASCII characters	D32, D16, or D8	VME source data width
format	ASCII characters	F32, I32, I16, I8, U32, U16, U8, or or Point	VME source data format
byte_order	ASCII character	M or I	M = Motorola (big endian) I = Intel (little endian)
count	integer - constant - variable Ix, I[x],GIx, GI[x] - label		number of data items to read
source	- variable Ix, I[x] or Fx,F[x] - global variable GIx,GI[x] or GFx,GF[x] - absolute point table ABS[n] - relative point table REL[n] - register Rx, R[x] - label		the starting address of the first item in the list of data to write

Variables or labels used for arguments must equate to valid values at run-time or an error will result.

Description:

VME/WRITE copies data, such as one or more of the integer or floating point variables, or absolute or relative point table entries, from the specified source CLC local memory to the specified VME memory address.

The instruction specifies of the number and width of the data items transferred. Proper word and double word byte order for 16 and 32 bit data sizes is controlled by the M or I (big or little endian) argument. Therefore, both Motorola and Intel architectures are easily supported.

Format specifiers:

F32	32-bit	floating	point
I32	32-bit		integer
I16	16-bit		integer
I8	8-bit		integer
U16	16-bit	unsigned	integer
U8	8-bit unsigned integer		

For more information on the limitations of direct VME memory accesses see the CLC/VME Direct Data Access section in the chapter on CLC/VME SYSTEMS.

Example:

```
VME/Write   A32, 0x402000, D16, U16, M, 10, I[01]
```

The contents of the CLC integer variables I[1] through and including I[10] are copied to the specified VME memory space. Since D16 is specified only the lower 16 bits of each 32 bit integer variable is written to VME memory. (Since the CLC uses a 680x0, M is specified and byte swapping is not required.)

8 Program Debugging and Monitoring

8.1 Finding Program Problems

Locating and eliminating program errors can be a frustrating effort. Therefore, design a program before you start writing it (or making modifications). Attempting to incrementally fix an ill conceived structure usually results in a complex collection of logic that eventually even the author doesn't understand.

Start with simple, basic program blocks. Test the blocks, independently if possible, even if testing requires writing a bit more program just for test purposes. A tested and dependable section of a program allows you to focus on just the potential problem areas. If the program compiles correctly, make sure that the problem lies with the program, not the hardware. If necessary, write short test programs to test individual hardware functions.

Use a program branch and the CLC's message capabilities to insert a message into your program. Momentarily stopping the program and checking critical values can tell you where things are going wrong.

Think through the implications of using triggered events. Remember that events and the execution of event functions typically occurs asynchronously to program tasks. You cannot always depend on the timing of triggered events. It may be necessary to add additional program code to provide synchronization.

The following Task parameters can also be used to help with program debugging:

T-0-0130	Current Instruction Pointer
T-0-0131	Current Instruction
T-0-0132	Instruction Pointer at Error
T-0-0133	Composite Instruction Pointer
T-0-0135	Current Subroutine
T-0-0136	Stack Variable Data
T-0-0137	Task Subroutine Breakpoint
T-0-0138	Sequencer Information
T-0-0200	Last Active Event Number

See Chapter 3. Parameters for more information.

See Appendix B - Communication Error Codes and Messages

If the program does not compile, or compiles with errors, use VisualMotion's "Display Code" selection from the File menu to check that the compiler is generating the instructions you intend. Remember that the compiler doesn't check your program's logic, the compiler can only check for proper syntax and use. VisualMotion's compilers typically provide error or warning dialog boxes that refer to line numbers in the displayed code. The following section provides the syntax of the code displayed.

Test Code

A typical example of additional code for testing program functionality is the use of counters. One way to implement a counter would be to change the state of an I/O bit after a distance event has occurred. After each move, which is supposed to trigger the event, increment an integer variable called “move_count”. Then use a branch statement to test whether the I/O bit did, in fact, change state. If it did, then increment an integer variable called “event_count”. The final value of “event_count” can be compared to “move_count” to see if, in fact, the event occurred once for every move.

8.2 CLC Compiler Base Code

Compiling an Icon or Text Language program produces a text file output listing in CLC Base Code, using mnemonics and syntax similar to assembly language. The Base Code resulting from the compilation of a program may be viewed using Window's Notepad by selecting "Display Code" from the VisualMotion File menu. Base Code may also be viewed using a compatible ASCII-only text editor.

Base Code is typically used as an aid to debugging when checking a program for logical errors. Base Code files are view-only program listing files. Editing a Base Code file has no effect on a subsequent re-compilation of the program.

The labels in a Base Code listing result from both user defined labels and the labels that are generated internally by the Visual Motion compiler.

Base Code instruction mnemonics and valid arguments

The following lists the Base Code instruction mnemonics and valid arguments. Instructions requiring more than one argument show the arguments separated by commas ",". Alternative forms for arguments are shown by enclosing a general form for each argument in square brackets, separated by a vertical bar "|".

ABORT_PATH

ABORT_PATH [task]

Halts coordinated motion in the specified task.

ACCEL

ACCEL [axis | label | Ix | GIx],[rate | label | Fx | Gfx]

Sets the acceleration rate for the specified axis.

AXES

AXES task mode, axis, axis, axis

Specifies the axes to be assigned to this task and how they will be used. (All axes used in a task must be declared for that task.)

task mode =
- 1 for single axis non-coordinated motion
- 2 for coordinated axis for multi-axis coordinated motion
- 3 for velocity mode, rotation only - no axis positioning
- 4 for ratioed slave axis
- 5 for ELS mode
- 6 for Torque mode

axis = a valid identifier for an axis, from 1 to the maximum number of axes

AXES_GROUP

AXES_GROUP task mode, axis, axis, axis, axis, axis, axis

AXES_GROUP

AXES_GROUP is similar to the AXES command; however it permits up to six axes.

AXIS_EVENT

AXIS_EVENT [axis | label | Ix | GIx], event #1, event #2, event #3, event #4

Enables up to four Repeating Position events for a single-axis, ELS, ratio, or velocity mode axis.

Event # = [an integer Event # | label | Ix | GIx]

AXIS_WAIT

AXIS_WAIT [axis | label | Ix | Gix]

If the argument is a positive integer representing a valid axis, program execution waits for the axis to be within its preset DDS position window. If the argument is -1, program execution waits until all axes in the task are within their position windows. The position window is defined by the DDS parameters: Position Window and Zero Velocity. AXIS_WAIT will wait indefinitely if used with velocity mode (axis task mode 3) since positioning is not used.

BNE

BNE label (subroutine or event)

Branches to label if the task's status word is set to "not equal".

BEQ

BEQ *label (subroutine or event)*

Branches to label if the task's status word is set to "equal".

BGT

BGT *label (subroutine or event)*

Branches to label if the task's status word is set to "greater than".

BLT

BLT *label (subroutine or event)*

Branches to label if the task's status word is set to "less than".

BGE

BGE *label (subroutine or event)*

Branches to label if the task's status word is set to "greater than or equal".

BLE

BLE *label (subroutine or event)*

Branches to label if the task's status word is set to "less than or equal".

BRA

BRA *label (subroutine or event)*

Branch to label, always (no matter what)

CALL_FUNC

CALL_FUNC *func_offset, ret_pointer, arg_count, arg1_ptr, ... argn_ptr*

Calls the function at func_offset with a return pointer and a variable number of arguments.

func_offset	offset in bytes from current program counter to start of function

ret_pointer	pointer to int or float return variable if (0), there is no return value
arg_count	number of arguments passed to the function if (0), there are no arguments there can be between 1 and 5 arguments
arg1_ptr... argn_ptr	pointer to argument passed to function can be int, float, global int, global float, constant int, constant float, local int, local float, absolute or relative point label used as initial value of local variable

CALC

CALC evaluates the equation

CAP_ENABLE

CAP_ENABLE axis, probe, event#

Enables the event on the axis for the probe transition. When the transition occurs, the event triggers.

axis = from 1 to the maximum number of axes

probe:

1 = probe 1, 0 -->1
2 = probe 1, 1 --> 0
3 = probe 2, 0 --> 1
4 = probe 2, 1 --> 0

event = [event # | label | Ix | GIx]

CAP_SETUP

CAP_SETUP axis, probe

At program activation the drive is configured to capture feedback position on it's probe transition and to include position data in its cyclic telegram data.

axis = from 1 to the maximum number of axes

probe:

1 = probe 1, 0 -->1
2 = probe 1, 1 --> 0
3 = probe 2, 0 --> 1
4 = probe 2, 1 --> 0

CLEAR

CLEAR [Ix | GIx | Fx | GFx | label]

Sets integer or float variable to zero

COMP

COMP *[Ix | GIx | Fx | GFx | label], [Ix | GIx | Fx | GFx | label]*

Set the task's status word to the logical result of 1st argument minus 2nd argument

DATA_SIZE

DATA_SIZE *I, F, ABS, REL, EVT, ZONE*

Sets the amount of memory allocated for each type of data in one of the four program tasks. (The total program requirement is the sum of the DATA_SIZE allocations for each task in the program.)

I = the number of integer variables allocated for this task
F = the number of floating point variables allocated for this task
ABS = the number of absolute point table entries allocated for this task
REL = the number of relative point table entries allocated for this task
EVT = the number of event table entries allocated for this task
ZONE = the number of zone table entries allocated for this task

DEC

DEC *[Ix | GIx | label]*

Subtracts 1 from the specified integer variable

DECEL

DECEL *axis, rate*

Sets the deceleration rate for the axis
axis = [integer constant | label | Ix | GIx]
rate = [floating point constant | label | Fx | GFx]

ELS_ADJUST

ELS_ADJUST *axis, offset*

Sets the phase or velocity offset for the ELS axis.
axis = [integer constant | label | Ix | GIx]
offset = [floating point constant | label | Fx | GFx]

ELS_INIT

ELS_INIT *els type, slave axis, master axis, encoder, sync type*

Initializes the relationship between master and slave axes.

els type:
1 = virtual master
2 = real master (daisy-chained)
3 = real master (SERCOS)
4 = follow axis feedback

slave axis = [integer constant | label]

master axis = [integer constant | label]

encoder:
1 = primary encoder
2 = secondary encoder

sync type:
1 = velocity
2 = phase

ELS_MODE

ELS_MODE axis, mode

Sets the mode for the specified ELS axis.

axis = [integer constant | label | Ix | GIx]

mode:
1 = single axis
2 = ELS synchronization

END

Defines the end of the program for this task.

EVENT_DONE

EVENT_DONE event

Marks the specified event status as complete.

event = [integer constant | Ix |GIx | label]

EVENT_ENABLE

EVENT_ENABLE event

Activates the specified repeating timer event.

event = [integer constant | Ix | GIx | label]

EVENT_END

Defines the end of an event routine program code.

EVENT_START

Marks the beginning of an event routine program code.

EVENT_WAIT

EVENT_WAIT event

Pauses task execution until the specified active event completes.
event = [integer constant | Ix | GIx | label]

FUNC_ARG

func_label: FUNC_ARG label, type, <min value>, <max value>

Declares local variables.

func_label	text label of function
label	text string identifier of local variable
type	‘F’=float, ‘I’=integer, “ABS”= ABS point index, “REL”= REL point index
min value	optional minimum value of argument
max value	optional maximum value of argument

FUNC_END

func_label: FUNC_END return value

Indicates the end of a function and optional return value.

func_label	text label of function
return value	return argument

FUNC_START

func_label: FUNC_START

Indicates the start of the function named by ‘func_label’.

func_label	text label of function

GET_PARAM

GET_PARAM type, set, ID number, destination

Copies the specified parameter data to the specified integer or floating point variable (the variable type must match the parameter type).

type:

A = axis
C = system
D = drive
T = task

set = axis or drive ([integer constant | Ix | label]), or task ID letter

ID number = identifying parameter number (range 1 to 65535)

destination = destination variable, [Ix | GIx| Fx | GFx | label]

GO

GO axis

Starts continuous motion on the axis. The axis must be configured as non-coordinated or velocity mode.

axis = [integer constant | Ix | GIx | label]

HOME

HOME axis

Enables motion homing the specified axis. (The homing parameters must have been set in the DDS drive.)

axis =[integer constant | Ix | GIx | label]

INC

INC [Ix | label]

Adds 1 to the specified integer variable.

LOCAL/VARIABLE

func_label: LOCAL/VARIABLE label, type

Declares local variables.

func_label	text label of function
label	text string identifier of local variable
type	'F'=float, 'I'=integer

MESSAGE_PORT

func_label MESSAGE_PORT target, string, <pointer>

Outputs formatted string to designated port.

func_label	text label of function
target	1 = diagnostic message. 2 = status message. 3 = serial host port(Port A). 4 = serial teach pendant port(Port B).
string	formatted text string to display, formatting types are %d, %f, %x
pointer	optional single argument - Rx, Fx, Ix, GFx , or GIx

MOVE_JOINT

MOVE_JOINT ABS point

Moves the joint based on an absolute point (Six-axis CLC only.)

ABS point = [integer constant | Ix | GIx | label], an entry in the absolute point table

KINEMATIC

KINEMATIC kinematic library number

Selects the set of equations specified by the library number from an optional kinematic library. Used to translate Cartesian coordinates for custom coordinated motion applications such as robots.

MOVEA_AXIS

MOVEA_AXIS axis, distance, event, event, event, event

Starts single axis absolute motion for the specified axis, and activates the specified events.

axis = [integer constant | Ix | GIx | label]

distance = [floating point constant | Fx | GFx | label]

event = [integer constant | Ix | GIx | label]

MOVER_AXIS

MOVER_AXIS axis, distance, event, event, event, event

Starts single axis relative motion for the specified axis, and activates the specified events. An event is specified by an integer number index into the event table or a label for an integer variable containing the index.

axis = [integer constant | Ix | GIx | label]

distance = [floating point constant | Fx | GFx | label]

event = [integer constant | Ix | GIx | label]

MOVEA_PATH

MOVEA_PATH ABS point

Starts coordinated motion from the current position to the point specified in the absolute point table.

ABS point = [integer constant | Ix | GIx | label]

MOVER_PATH

MOVER_PATH REL point, ABS point

Starts coordinated straight line motion from the current position to the point specified by the vector sum of the absolute and relative points.

ABS point = [integer constant | Ix | GIx | label]
REL point = [integer | Ix | GIx | label]

MOVEA_CIRCLE

MOVEA_CIRCLE ABS point, ABS point

Starts coordinated motion from the current position, through the first specified point, ending at the second specified point.

ABS point = [integer constant | Ix | GIx | label]

MOVER_CIRCLE

MOVER_CIRCLE REL point, REL point, ABS point

Starts coordinated motion from the current position, through the point specified by the vector sum of the ABS point and the first REL point, ending at the point specified by the vector sum of the ABS point and the second REL point.

ABS point = [integer constant | Ix | label]
REL point = [integer constant | Ix | label]

MSG_DIAG

MSG_DIAG ASCII text string

Sets the current diagnostic message to the specified ASCII text string.

MSG_STATUS

MSG_STATUS ASCII text string

I
Sets the current status message to the specified ASCII text string.

PARAM_BIT

PARAM_BIT type, set, ID number, source, I/O mask

Sets the parameter bit specified by the type, set, ID number and I/O mask to the value in the specified source variable at initialization.

type:

A = axis
C = system
D = drive
T = [A | B | C | D] (task ID letter)

set = [integer constant | Ix | GIx | label] for axis or drive; or [A | B | C | D] for task

ID number = [integer constant] for a parameter number in the range 1 to 65535

source = [integer constant | floating point constant | Ix | GIx | Fx | GFx | label]

I/O mask = specifies 1 to 16 bits in an I/O register

PARAM_INIT

PARAM_INIT type, set, ID number, source

Sets the specified parameter to the value in the specified variable at initialization.

type:

A = axis
C = system
D = drive
T = [A | B | C | D] (task ID letter)

set = [integer constant | Ix | GIx | label] for axis or drive; or [A | B | C | D] for task

ID number = [integer constant] for a parameter number in the range 1 to 65535

source = [integer constant | floating point constant | Ix | GIx | Fx | GFx | label]

PID_CONFIG

PID_CONFIG #, type, control_register, status_register, loop_time, set_point_type, set_point, set_point_axis, feedback_type, feedback, feedback_axis, output_type, output, output_axis, control_block

#: PID loop number, range 1-10.

type: PID loop type, currently only 1 is valid.

control_register: label or number of register used for control of this loop.

status_register: label or number of register used for status of this loop

loop_time: update time of this loop, multiples of 8 millisecond.

set_point_type: Type of set point, 1=variable, 3=unsigned register, 4=signed register

set_point: Axis parameter, register, variable, or equivalent label to be used as the set point of this loop.

set_point_axis: For axis parameters, axis number; else 0.

feedback_type: Type of feedback, 1=variable, 2=axis parameter, 3=unsigned register, 4=signed register

feedback: Axis parameter, register, variable, or equivalent label to be used as the feedback of this loop.

feedback_axis: For axis parameters, axis number; else 0.

output_type: Type of output, 1=variable, 2=axis parameter, 3=unsigned register, 4=signed register

output: Axis parameter, register, variable, or equivalent label to be used as the output of this loop.

output_axis: For axis parameters, axis number; else 0.

control_block: First variable in a block of 20 float variables(Fx) to be used for this loop.

See also **VAR_INIT**.

POSITION

POSITION *task, ABS point*

Copies the current position coordinates of the specified task to the specified ABS point table entry. The contents of the point table entry are overwritten.

task = [A | B | C | D]

ABS point = [integer constant | Ix | GIx | label]

RATIO

RATIO *master axis, slave axis, master ratio, slave ratio*

Sets the ratio between the specified master and slave axes.

master axis = [integer constant | Ix | GIx]

slave axis = [integer constant | Ix | GIx]

master ratio = [floating point constant | Fx | GFx]

slave ratio = [floating point constant | Fx | GFx]

READ

READ *register, count, target variable*

Copies the contents of the specified I/O register(s) to the lower 16 bits of the specified integer variable(s). The upper word of the variable(s) are zero-filled. Only a contiguous block of registers can be moved.

register = an integer constant specifying the number of the starting source I/O register
count = a positive integer constant for the number of register to copy
target variable = the starting integer variable table entry for the destination of the data

RESUME_PATH

RESUME_PATH *task*

Restarts previously halted coordinated motion in the specified task.

task = [A | B | C | D]

RETURN

Marks the end of a subroutine's program code, and returns program execution to the calling program.

SET

SET I/O state, register, I/O mask

Sets the specified register's bits, that are enabled by the I/O mask, to the state specified by I/O state.

I/O state = 16 bit binary word of bits to set in the specified register. 0 = off, 1 = on.

register = an integer number specifying an I/O register

I/O mask = 16 bit binary word specifying the bits that may be changed. 1 = enabled

SET_PARAM

SET_PARAM type, set, ID number, source

Copies the specified parameter's value to the specified integer or floating point variable. The source variable data type must match the destination parameter data type.

type:

A = axis
C = system
D = drive
T = task

set = [Ix | GIx | label] for axis or drive; or [A | B | C | D] for task ID letter]

ID number = identifying parameter number, within the range: 1 to 65535

source = [integer constant | floating point constant | Ix | GIx | Fx | GFx | label]

START

Marks the beginning of a task or subroutine.

STOP

STOP axis

Signals the DDS to halt single-axis or velocity mode motion on the specified axis if the argument is a positive integer (1 - 8). If the argument is -1, motion is halted for all single-axis and velocity mode axes in the task. Signaling the DDS to halt motion decelerates the axis to zero velocity using the deceleration rate programmed in the appropriate DDS drive parameter.

axis = [integer constant | Ix |GIx | label]

STOP_PATH

STOP_PATH task

Stops coordinated motion in the specified task.

task = [A | B | C | D]

TEST

TEST register, I/O mask

Sets the task's status word to the result of a logical AND of the specified register and the I/O mask.

register = a positive integer constant for a modifiable CLC register

I/O mask = 16 bit binary word

VAR_INIT

VAR_INIT ar_start, arg1, arg2, arg3,....., arg20

var_start: First variable in a block of program variables(Fx, Ix) to be initialized.

arg1- arg20: values to initialize with.

VEL

VEL axis, rate

Sets the velocity specified by rate in the specified task axis.

axis = [integer constant | Ix | GIx | label]

rate = [floating point constant | Fx | GFx | label]

VME_EVENT

VME_EVENT card, priority, event

Sends a broadcast event across the VME backplane to cards selected with the specified priority and event number.

card = CLC unit as a power of 2 (i.e. unit 1 = 2, unit 3 = 8), or 0xFFFF for all cards

priority = [A | B | C | D] (task ID letter)

event= [integer constant|Ix|GIx|label], the lesser of 64 or the maximum number of events

VME_READ

VME_READ *[Ax],[address],[Dxx],[format],[order],[count],[target]*

Reads one or more VME memory locations to the specified target register(s).

Axx = [A32 | A24 | A16], the VME source memory address width

address = the VME memory address of the source data, in hexadecimal (Depends on the source address width.)

Dxx = [D32 | D16 | D8], the VME memory source data width

format = [F32 | I32 | I16 | I8 | U16 | U8], the format of the VME source data

order = [M | I] (for Motorola or Intel), the VME source data byte order

count = [integer constant | Ix | GIx | label], the number of locations to read

target = [Ix| GIx | Fx |GFx | label], the beginning CLC table variable to receive the data

VME_WRITE

VME_WRITE *[Axx],[address],[Dxx],[format],[order],[count],[source]*

Writes to one or more VME memory locations from a specified source register(s).

Axx = [A32 | A24 | A16], the VME target memory address width

address = the VME memory address of the target data, in hexadecimal (Depends on the source address width.)

Dxx = [D32 | D16 | D8], the VME memory target data width

format = [F32 | I32 | I16 | I8 | U16 | U8], the format of the VME source data

order = [M | I] (Motorola or Intel), the VME target data byte order

count = [integer constant | Ix |GIx | label], the number of locations to read

source = [Ix |GIx | Fx |GFx | label], the beginning CLC table variable to read to VME memory

WAIT

WAIT *time*

Suspends task execution for the number of milliseconds specified by time.

time = [integer constant | Ix | GIx | label]

WAIT_IO

WAIT_IO *register, I/O mask, I/O state*

Suspends task execution until the specified I/O conditions are met.

register = an integer constant for a CLC register

I/O mask = identifies 1 to 16 bits in an I/O register

I/O state = 0 --> off; non-zero --> on

WAIT_PATH

WAIT_PATH task, ABS or REL point, condition

Suspends task execution until the specified path planner conditions are met.

task = [A | B | C | D]

ABS point = [integer constant | Ix | GIx | label], reference to an absolute table entry
REL point = [integer constant | Ix | GIx | label], reference to a relative table entry

condition:

0 = Ready
1 = Accel
2 = Slew
3 = Blending
4 = Target decel
5 = Controlled stop
6 = Stopped
7 = At target
8 = Done

WRITE

WRITE register, count, source

Copies the data in the specified integer variable(s) to the specified I/O register.

register = an integer number for the starting destination I/O register

count = a positive integer constant for the number of registers to copy

source = an integer number for the starting source integer variable table entry

8.3 Icon Language Warnings and Error Messages

The following warning messages are generated by the VisualMotion Icon Compiler. After receiving a warning message you may continue or exit the compilation.

Data missing in one or more fields, do you still wish to continue?

Caution! Changing Modes may halt motion. Continue?

Caution! Changing Modes may start motion. Continue?

The following error messages are displayed by VisualMotion's Icon Compiler.

Valid Entries are '0' or '1'.

Invalid name!

Cannot change task or open dialog box while dialog box is open

Axis undefined or not unique.

Valid event numbers are 1 to 100.

Valid axis numbers are 1 to 8.

Valid number range is 1 - 32767.

Valid percents are 1 - 100.

Labels must start with an alpha character!

Label name already exists!

Number missing or out of range.

Selected Icon is not a subroutine or no icon selected!

Data Field Empty!

Label type must be defined!

Task name undefined.

No filename specified.

Non-Branch icons have only two output connections.

Branch icons have only two output connections.

Point out of range.

Connection could not be made, try connecting adjacent ---?

Only connections between icons or adjacent blocks can be ---?

Finish icon not found or open path!

Start icon not found or multiple Start icons found!

Icon program not found!

Cannot open code file!

Unknown icon term _____.

Missing axis selection.

Open in program flow, at or near highlighted icon, ---?

Branch Icon has missing connection or one in wrong dire ---?

No axis selected!

Time Delay out of range!

Could not initialize update timer!

Operation type not selected!

Drive numbers doesn't match.

Should drive number be c ---?

Can't open file _____!
Source or target not selected!

Valid range ___ - ___

Valid range ___ - ___

CLC card parameters cannot be changed!

File syntax other than parameters!

File of different type parameters!

CLC card is not communicating!

No selection made!

8.4 Text Language Error Messages

The following are error messages produced by the Visual Motion text compiler. Line numbers refer to code displayed by selecting "Display Code" from the Visual Motion ***File menu***. For further information on the format of the code displayed see CLC Compiler Base Code.

First Pass Errors

CLC code converter error log file!

Unable to open source file!
Line [nnn], Maximum number of terms reached!
Line [nnn], unknown mnemonic operator - [xxx]
Line [nnn], unknown, missing or wrong argument - [xxx]
Line [nnn], missing point argument!
Line [nnn], missing closing bracket "]"!
Line [nnn], additional arguments - [xxx...]!
Line [nnn], point number '??' out of range (1-nn)!
Line [nnn], missing arguments!
Line [nnn], unknown IF conditional terms - [?] [?]!
Line [nnn], ELSE or ENDIF without IF term!
Line [nnn], maximum number of nested IFs exceeded!
Line [nnn], sequencing error, IF, ELSE, or ENDIF imbalanced
Line [nnn], missing message text!
Line [nnn], incompatible circle arguments - [xxx]
Line [nnn], variable out of range - [variable name]
Line [nnn], right side of EQU must be a number - [____]!
Line [nnn], label [label name] not found
Line [nnn], arguments must be integer or constant!
Line [nnn], bit number [nn] out of range (1-nn)!
Line [nnn], register number [nn] out of range (1-nn)!
Line [nnn], integer variable number [nn] out of range (1-nn)!
Line [nnn], register number + count exceed range (1-nn)!
Line [nnn], axis number [nn] out of range (1-n)!
Line [nnn], mode number [nn] out of range (0-n)!
Line [nnn], mark "_____" also defined on line [nn]!
Mark [_____] on line [nnn] was not referenced in program!
Mark [_____] used on line [nnn] is not declared!
Line [nnn], event number [nn] out of range (1-nn)!
Line [nnn], delay value [n...] out of range (1-n...)!
Line [nnn], too many arguments!

Second Pass Compiler Errors

'xxxx' - unresolved mark reference.
Line xx, all probe types zero or not unique!
Line xx, argument 'yyyy' out of range!
Line xx, axes missing or not unique!
Line xx, axis number 'yyyy' out of range(wwww,yyyy,1-zzzz).
Line xx, bit number 'yyyy' out of range(1-16)!

Line xx, 'compare' arguments must be floats, integers, or constants!
Line xx, event element 'yyyy' missing or entered with spaces!
Line xx, event element 'yyyy' unknown!
Line xx, event EVT[].yy data is not changeable in program!
Line xx, event function 'yyyy' not found in program!
Line xx, event message 'yyyy' must start with quotes!
Line xx, event number 'yyyy' out of range!
Line xx, float number 'yyyy' conversion error!
Line xx, hex number 'yyyy' conversion error!
Line xx, integer number 'yyyy' conversion error!
Line xx, invalid argument 'yyyy'!
Line xx, invalid cam number 'yyyy'! Range 1 to 8.
Line xx, invalid count or count plus register exceeds range!
Line xx, invalid Encoder type 'yyyy', 1=primary, 2=secondary!
Line xx, invalid ELS type 'yyyy', range 1 to 4!
Line xx, invalid sync type 'yyyy', 1=velocity, 2=phase, 3=cam!
Line xx, invalid VME Address 'yyyy'!
Line xx, invalid VME address width 'yyyy'!
Line xx, invalid VME byte order 'yyyy'!
Line xx, invalid VME count 'yyyy'!
Line xx, invalid VME data width 'yyyy'!
Line xx, invalid VME data format 'yyyy'!
Line xx, left term 'yyyy' of equation must not be constant!
Line xx, maximum number of terms reached.
Line xx, maximum size(20) of term exceeded!
Line xx, message exceeds 80 characters!
Line xx, missing argument(s)!
Line xx, missing beginning quotes of message!
Line xx, missing closing bracket ']'!
Line xx, missing closing curly brace '}'!
Line xx, missing closing quotes of message!
Line xx, missing mark name!
Line xx, Parameter <type> must be 'A', 'C', 'D' or 'T'
Line xx, point element 'yyyy' missing or entered with spaces!
Line xx, point element 'yyyy' unknown!
Line xx, register number 'yyyy' out of range(1-zzzz)!
Line xx, table or array index out of range 'yyyy'!
Line xx, table or array label index out of range 'yyyy'!
Line xx, Task must be 'A', 'B', 'C' or 'D'!
Line xx, too many arguments!
Line xx, unknown mnemonic operator - 'yyyy'!
Line xx, unknown or out of range variable 'yyyy'!
Line xx, unresolved index 'yyyy'!

Line xx, unresolved index label 'yyyy'!

Line xx, unsupported data transfer! VME bus width 'yyyy', VME format 'zzzz', local

Line xx, unsupported structure transfer!

Line xx, valid modes are 1=single axis, 2=ELS synchronized!

Line xx, variable table 'yyyy' index unknown!

Mark table filled - yyyy, reduce number of subroutine calls.

No main task(A, B, C, or D) found!

Sequencing error in output file!

Size of program exceeds compiler space!

Unable to allocate memory for compiler!

Unable to open source file.

Write to file error!

'*xxxx*' - unresolved mark reference.

The mark '*xxxx*' was used as a destination in a branch or subroutine call, but was not found in the code. Check for possible spelling error or missing subroutine.

Line *xx*, all probe types zero or not unique!

The probe arguments are both zero or are the same.

Line *xx*, argument '*yyyy*' out of range!

The argument *'yyyy'* is out of range, check syntax in manual.

Line *xx*, axes missing or not unique!

In a AXES_GROUP command for ratioed axis, the slave axis argument is zero or is the same as the master axis.

Line *xx*, axis number '*yyyy*' out of range(*wwww*,*xxxx*,1-*zzzz*).

The axis number or label '*yyyy*' has not been resolved to a valid number. The numbers '*wwww*', '*xxxx*', and range 1 to '*zzzz*' are valid axis numbers.

Line *xx*, bit number '*yyyy*' out of range(1-16)!

On line '*xx*', the string '*yyyy*' is evaluated to number outside of the valid range for register bits.

Line *xx*, 'compare' arguments must be floats, integers, or constants!

Compare arguments must be Fx, GFx, GIx, Ix or equivalent labels or constants. Compares are derived from "IF" statements in textual language programs or "BRANCH" icons in GUI programs.

Line *xx*, event element '*yyyy*' missing or entered with spaces!

On line 'xx', the compiler has not found a "]" in the event string 'yyyy'. It uses this to position to the start of the event element. The event element { s, t, d, a, f, m }must follow immediately.

Line *xx*, event element '*yyyy*' unknown!

The event element '*yyyy*' was not found in the event element table, check manual for exact syntax.

Line *xx*, event EVT[].*yy* data is not changeable in program

Line *xx*, event function '*yyyy*' not found in program!

The event function 'yyyy' was not found in the program. Check spelling and capitalization.

Line *xx*, event message '*yyyy*' must start with quotes!

The compiler is expecting a quote to start the ASCII string for the event message, but did not find it.

Line *xx*, event number '*yyyy*' out of range!

On line '*xx*', the string '*yyyy*' was evaluated to be out of the range for events defined for this program. Events and other variables are declared in the "DATA/SIZE" command in a textual language program or by the "SIZE" icon in GUI programs.

Line *xx*, float number '*yyyy*' conversion error!

The string 'yyyy' for conversion to a float was determined to contain one of the following errors:

No numeric characters.

More than one exponent symbol 'E'('e').

More than two sign symbols'.

More than one decimal point.

Alpha characters other than 'E'('e').

Line *xx*, hex number '*yyyy*' conversion error!

On line '*xx*', the string '*yyyy*' is greater than 10 characters long or contains non-hexadecimal characters. Valid strings start with 0x and contain ASCII characters 0-9, A-F or a-f (0x1BF8).

Line *xx*, integer number '*yyyy*' conversion error!

The string 'yyyy' for conversion to an integer was determined to contain one of the following errors:

No numeric characters.

Number of numeric characters exceed 10.

The converted number exceeds 0x7FFFFFFF.

Line *xx*, Invalid argument '*yyyy*'!

Line *xx*, Invalid cam number '*yyyy*'! Range 1 to 8.

The CAM number *'yyyy'* was evaluated to be less than one or greater than 8.

Line *xx*, Invalid count or count plus register exceeds range!

The count of registers to be transferred was evaluated to be less than one or when added to the starting register exceeds the maximum register range (1024 registers for CLC/V, 512 registers for CLC/D and CLC/P).

Line *xx*, Invalid Encoder type '*yyyy*', 1=primary, 2=secondary!

The ELS master encoder type *'yyyy'* was evaluated to be less than one or greater than 2.

Line *xx*, Invalid ELS type '*yyyy*', range 1 to 4!

The ELS type *'yyyy'* was evaluated to be less than one or greater than 4.

Line *xx*, Invalid sync type '*yyyy*', 1=velocity, 2=phase, 3=cam!

The ELS sync type *'yyyy'* was evaluated to be less than one or greater than 3.

Line *xx*, Invalid VME Address '*yyyy*'!

The VME address *'yyyy'* was evaluated to be less than one or greater than 0xFCFEFFFF.

Line *xx*, Invalid VME address width '*yyyy*'!

The address width *'yyyy'* was not found in the table of VME address widths.

{ "A16", "A24", "A32"}

Line *xx*, Invalid VME byte order '*yyyy*'!

VME byte order *'yyyy'* must start with 'I' or 'M', 'I' is for Intel order, 'M' is for Motorola. It can be a single character or the name, Intel or Motorola. It is case sensitive, so 'I' and 'M' must be capitalized.

Line *xx*, Invalid VME count '*yyyy*'!

The count of VME objects to transfer *'yyyy'* was evaluated to be less than one or greater than 32767.

Line *xx*, Invalid VME data width '*yyyy*'!

The data width *'yyyy'* was not found in the table of VME bus widths.

{ "D32", "D16", "D8"}

Line *xx*, Invalid VME data format '*yyyy*'!

The data format *'yyyy'* was not found in the table of VME data formats.

{ "I32", "I16", "I8", "U32", "U16", "U8", "F32", "POINT" }

Line *xx*, Left term '*yyyy*' of equation must not be constant!

A calculation must have a variable(Fx, GFx, GIx, Ix) or changeable table element(ABS[1].x, EVT[3].d, etc.) as its term to the left of the equal sign.

Line *xx*, Maximum number of terms reached.

When parsing the line 'xx', the number of terms exceeded 32. A term is one or more alphanumeric characters followed by a space, comma or other non-alphanumeric character. This error usually only occurs in message statements with many short words. Try a message with fewer words.

Line *xx*, Maximum size(20) of term exceeded!

While parsing line *'xx'* for arguments a string of more than 20 characters was encountered. Arguments and argument labels are limited to 20 characters. Check label length and use of commas between arguments.

Line *xx*, Message exceeds 80 characters!

The number of characters used in the message exceeds 80 characters. This count includes spaces.

Line *xx*, missing argument(s)!

One or more additional arguments were expected.

Line *xx*, missing beginning quotes of message!

On line 'xx', quotes were expected to denote the start of the message. Diagnostic, status and event messages are specified within quotes in textual

language programs. Also use quotes when using the "CALC" icon to set an event message.

Line *xx*, missing closing bracket ']'!

The closing bracket used to denote the end of the index of a data structure, was not found.

Line *xx*, missing closing curly brace '}'!

The closing brace used to denote the end of initialization data for a data structure was not found. Other causes are extra arguments or the wrong character.

Line *xx*, missing closing quotes of message!

On line 'xx', quotes were expected to denote the end of the message. Diagnostic, status and event messages are specified within quotes in textual language programs. Also use quotes when using the "CALC" icon to set an event message.

Line *xx*, missing mark name!

The argument of branch command does not start with an alpha character. Check for missing or misspelled argument.

Line *xx*, Parameter <type> must be 'A', 'C', 'D' or 'T'

The parameter class was not found to be 'A', 'C', 'D', 'T', or equivalent label. Check for missing or misspelled argument.

Line *xx*, point element '*yyyy*' missing or entered with spaces!

On line 'xx', the compiler has not found a "]." in the point string 'yyyy'. It uses this to position to the start of the point element. The point element { x, y, z, b, s, a, d, j, e1, e2, e3, e4, r, p, ya, el }must follow immediately.

Line *xx*, point element '*yyyy*' unknown!

The point element '*yyyy*' was not found in the point element table, check manual for exact syntax.

Line *xx*, register number '*yyyy*' out of range(1-*zzzz*)!

The register number '*yyyy*' is less than one or greater than the maximum register '*zzzz*'.

Line *xx*, table or array index out of range '*yyyy*'!

The table or array index '*yyyy*' is less than one or greater than number declared by DATA/SIZE command or by the default declaration.

Line *xx*, table or array label index out of range '*yyyy*'!

The table or array index label '*yyyy*' is evaluated to be less than one or greater than number declared by DATA/SIZE command or by the default declaration.

Line *xx*, Task must be 'A', 'B', 'C' or 'D'!

The compiler is expecting a task argument(A, B, C, or D) and has not found it. This may result from a missing argument or arguments out of sequence.

Line *xx*, too many arguments!

More terms than expected were found following the command. Check for extra arguments, extra commas or terms with spaces in them.

Line *xx*, unknown mnemonic operator - '*yyyy*'!

On line '*xx*', the string '*yyyy*' is assumed to be a command, but was not found in the list of valid commands. This error is most often generated from textual language programs when the command is misspelled or from incorrect syntax.

Line *xx*, unknown or out of range variable '*yyyy*'!

On line '*xx*', the string '*yyyy*' is not of the type expected. Check for argument type(float where integer should be used, etc.) , or for missing or misspelled arguments.

Line *xx*, unresolved index '*yyyy*'!

The index '*yyyy*' could not be resolved, check for missing or misspelled label. Label are case sensitive and cannot contain spaces.

Line *xx*, unresolved index label '*yyyy*'!

The index label '*yyyy*' could not be resolved to an integer or integer variable, check for missing or misspelled label. Label are case sensitive and cannot contain spaces.

Line *xx*, unsupported data transfer! VME bus width '*yyyy*', VME format '*zzzz*', local

The selected VME data transfer is not supported in this product. Check VME format and local format for possible erroneous selection.

Line *xx*, Unsupported structure transfer!

The data structures equated to each other are not of the same type. The data structure transfers supported are: Point to Point, Event to Event, and Zone to Zone.

Line *xx*, Valid modes are 1=single axis, 2=ELS synchronized!

The second argument of the "ELS/MODE" command is missing or out of range. This can also be generated if the first argument is invalid and appears as two or more arguments to the compiler.

Line *xx*, variable table '*yyyy*' index unknown!

The closing bracket is missing or other delimiter found in the index term of a variable or register with index format.

Mark table filled - *yyyy*, reduce number of subroutine calls.

The total number of marks used exceed the table space provided. Marks are the location tags of the start of tasks, event functions and subroutines, or, the destination of a branch or goto. Try to optimize your program to reduce the number of branches. If the problem persist, contact your Rexroth Indramat representative.

Upon successful completion of the compile, the number of marks and labels used are displayed in the completion window.

No main task(A, B, C, or D) found!

After compiling the program, no marks were found for Task_A, Task_B, Task_C, or Task_D. One or more of these task marks must be used. If it's a textual language program, check spelling and the underscore. The marks for the main tasks are case insensitive.

Sequencing error in output file!

While computing byte offsets for branches and subroutine calls, an unknown command op-code was encountered. This error can occur in a corrupted Windows memory system or a compiler bug. Try rebooting your computer and compiling again. If the problem persist, contact your Rexroth Indramat representative.

Size of program exceeds compiler space!

The compiler has 48k of space available for program development, this error is outputted when that space is filled. Variables and tables are not included in this space. Try reworking your program to fit it in the space.

Unable to allocate memory for compiler!

The 2nd pass compiler uses a large block of memory(48K) allocated from the Windows operating system to build the program. When Windows fails to

allocate this memory, this error is outputted. Try closing other applications or rebooting Windows to free needed memory.

Unable to open source file.

This error is issued on failing to open the file "CLCCODE.TXT". Some possible causes are:

File "CLCCODE.TXT" is not in the "\CLC" directory. This file is created by compiling a textual or icon program.

The maximum number of file are already open. DOS file "CONFIG.SYS" configures the maximum number of files.

The file is already open and cannot be shared.

Write to file error!

This error is outputted when the number of bytes sent to the output file doesn't match the number of bytes written in the output file. Check for available hard drive disk space or write protection on the output file(\clc\progrm*.exc).

A Direct ASCII Communication

A.1 Overview

The Indramat CLC / TRANS 01-D Motion Control can send and receive drive parameters, system parameters, user programs, and tables through its serial port. By using the text-based protocol described in this section, a wide variety of devices and programs can communicate with the CLC / TRANS 01-D. The protocol also supports ASCII PC ISA/ESIA and VME bus communication.

A.2 CLC Communication Protocol

The ASCII protocol format for the CLC / TRANS 01-D is designed so that all serial transmissions are similar in structure, facilitating simple coding/decoding routines. The protocol is the same for sending or receiving so that data may be easily handled and tracked.

All serial communications use the following standard protocol template. Information is classified using a three level system: command class, subclass, and data identifier.

```
>1 CS s.n 0123...CDEF $cs\r\n
|||||||||| |         |  | |
|||||||||| |         |  | |_ End of Message (i.e., cr/lf)
|||||||||| |         |  |_  Two Digit ASCII Checksum
|||||||||| |_________|_ Variable Length Data Field
|||||||||_ Number Within Set Data Identifier
||||||||_ ASCII period (0x2E)
|||||||_ Set Type Data Identifier
||||||_ optional ASCII space character (0x20)
|||||_ Command Subclass
||||_ Command Class
|||_ required ASCII space (0x20)
||_ Network Address (i.e. card position identifier)
|_ Start Character (0x3E)
```

Start Character The beginning of each message is identified by the ">" character (ASCII 62 decimal, 3E hex).

Network Address The network address can be used to support data transfer across a bus, or communication system, to multiple CLC / TRANS 01-D cards. The network address must be followed by an ASCII space character (0x20). An ASCII space (" ") character may be used to address the VME card containing the Host Communication serial port.

Data Identifier The Data Identifier field is a variable length field used to identify the data set being sent or requested. This identifier is in the format "s.n" or "s.n.x", where; "s"= the set identifier, may be a program handle, drive number, task ID, etc; and "n" = the numeric identifier used for items such as parameter numbers, table indexes and register numbers. The third field, called a "Step x" identifier, is used for parameter lists. No white space is allowed between the identifiers and the separating dot operators.

Variable Length Data This field contains the actual data being sent to or received from the CLC / TRANS 01-D. All subclasses have read and write capability. The same strings can be used for both responses and downloads.

Reading Data from the CLC / TRANS 01-D

To read data, the data portion of the string sent from the Host is empty.

Example:

The Host system requests a Drive Status Message

```
>1 DP 1.95 \r\n
           |_ No data sent, requesting current data
```

The CLC / TRANS 01-D responds with the current status of the specified drive:

```
>1 DP 1.95 302 Position Mode Encoder 1 c$\r\n
           |_ Data (status message)
```

Writing Data to the CLC / TRANS 01-D

To write data, send data in the data portion with the same starting protocol. Data received from the CLC / TRANS 01-D can be sent in the same format. The CLC / TRANS 01-D responds with an acknowledgment or an error message.

Example:

The Host sends the CLC / TRANS 01-D the DDS drive Kv Parameter for drive 1:

```
>1 DP 1.104 1.00$cs\r\n
            |_ New data sent
```

The CLC / TRANS 01-D has successfully accepted the parameter, since no error message was returned:

```
>1 DP 1.104 $cs\r\n
          |_ No message: data stored successfully
```

Communication Errors

If there is a checksum error, a format error, or an error in the data sent to the CLC / TRANS 01-D or the drive, the CLC / TRANS 01-D returns an error string in the data field. The string begins with a "!" character, followed by an error code and a descriptive message. Communication error codes and messages are listed at the end of this section.

Example:

```
>1 DP 1.104 !05 Greater than maximum value c$\r\n
            ||  |_ Error message
            ||_ Error Code (decimal)
            |_ Error indicator "!"
```

Checksum

A CLC / TRANS 01-D checksum is sent as two ASCII hexadecimal digits preceded by an ASCII '$'. The checksum is optional when requesting data from the CLC / TRANS 01-D. When sending data to the CLC / TRANS 01-D the checksum is required, unless it is disabled in the checksum parameter.

To compute the checksum, add the hexadecimal ASCII values of all of the characters before the '$', including the starting ">" character. Then add the most significant digit of the checksum to the two least significant digits. Negate ('+/-' on a calculator) this value to form the two's complement. The checksum transmitted to the CLC / TRANS 01-D is the 2 least significant digits of this number. This is the same checksum method used with the Indramat CLM Positioning Control and Station Operator Terminal (SOT).

Example:

Checksum on >1 AP 1.1 2

```
 3E hex
 31
 20
 41
 50
 20
 31
 2E
 31
 20
+32
------
222     (sum)

 22  (two least significant digits)
+ 2  (add two)
------
 24  (two's complement)Checksum=24(DC in hexadecimal)
```

End of Message

An ASCII carriage return (CR = 13 in decimal or 0d hexidecimal) and linefeed (LF, 10 decimal or 0a hex) combination is used for terminal compatibility. This document uses the notation '\r'(return) and '\n'(newline), as used with the C programming language, interchangeably with the CR LF notation. The CLC / TRANS 01-D always sends a CR LF combination, but will accept either a single LF or a CR LF from the Host device.

Backspaces and White Spaces

The ASCII backspace character (8 decimal or hex) erases the previous character from the CLC / TRANS 01-D's serial buffer except at the start of a message. This is useful for editing strings entered at a terminal. Also, any whitespace character (tab or space) can be used as a delimiter in strings. Whitespaces between fields or at the end of a message are discarded by the CLC / TRANS 01-D.

Numeric Data Formats

The CLC / TRANS 01-D sends numeric data in ASCII parameter-specified units and scaling format. The format of floating point data depends on the data's use and how and where it is stored. Floating point data of fixed precision (e.g., drive data) uses fixed resolution. The resolution of data stored on the CLC / TRANS 01-D card (i.e., local or global variables) depends on the storage precision used; 32 or 64-bit.

Floating point data that is too large or small to be printed in decimal format is represented in scientific notation. *Hexadecimal data* is sent and received with an '0x' prefix. *Binary data* is represented as a 16 digit string of ASCII "1" or "0" characters.

Example:

- ***Floating point position data**:* 0.0100 123.4567 -12.0000 12.3e+16 *(resolution = 0.0001 units)*
- ***Integer data**:* 0 1000 -10
- ***Hexadecimal data:*** 0x12AB 0x1234ABCD
- ***Binary data:*** 0000111100001111

Format of Data Sent to the CLC / TRANS 01-D

Any resolution can be used for data sent to the CLC / TRANS 01-D. Numbers may be padded with zeros or spaces at either end as a visual formatting aid when entering data from a terminal. Padding applies to data identifiers as well as the data field.

The resolution of the data stored on the CLC / TRANS 01-D is the resolution of the data the CLC / TRANS 01-D sends to the Host on request. Floating point numbers may also be sent in scientific notation.

A.3 Command Classes/Subclasses

The tables below list the command class identifier and the subclass identifier for each of the available subclasses within the command class.

Parameters

Command Class	Command Subclass
A - CLC / TRANS 01-D Axis Parameters	A - Attributes
	B - Block List Parameter
	D - Lists/Tables
	H - Upper Limit
	L - Lower Limit
	P - Parameter Data
	T - Name Text
	U - Units Text
C - CLC / TRANS 01-D Card Parameters	(Same subclasses as CLC / TRANS 01-D Axis Parameters)
D - SERCOS Drive Parameters	(Same subclasses as CLC / TRANS 01-D Axis Parameters)
T - Task Parameters	(Same subclasses as CLC / TRANS 01-D Axis Parameters)

Variables

Command Class	Command Subclass
I - Integer Variables	P - Data
	T - Text Label
F - Floating Point Variables	(Same subclasses as Integer Variables)
G - Global Integer Variables	(Same subclasses as Integer Variables)
H - Global Floating Point Variables	(Same subclasses as Integer Variables)

PID

Command Class	Command Subclass
M - PID loop	B - Set Point variable, Feedback variable, Output variable, First variable of control block
	E - Calculated set point
	F - Calculated feedback
	G - Calculated output
	J - Loop time
	L - List of valid PID loop numbers
	R - Control register
	S - Status register
	T - Type

Point Tables

Command Class	Command Subclass
X - Absolute Point Table	X - X coordinate
	Y - Y coordinate
	Z - Z coordinate
	B - Blend Radius
	S - % of Max Speed
	A - % of Max Acceleration
	D - % of Max Deceleration
	J - Jerk limiting %
	1 - Event 1
	2 - Event 2
	3 - Event 3
	4 - Event 4
	R - Roll (6-axis robot)
	R - Rate (Kinematic 8 only)
	P - Pitch (6-axis robot)
	W - Yaw (6-axis robot)
	E - Elbow (6-axis robot)
	V - List of all user defined labels
	L - List all of the above in one row
Y - Relative Point Table	(Same subclasses as X Absolute Point Table)

Event Tables

Command Class	Command Subclass
E - Event Table	A - Argument
	D - Direction
	F - Function
	L - List all elements in one row
	M - Message
	S - Status
	T - Type

Program Communication

Command Class	Command Subclass
B - TRANS 01-D	A - CTA 10 Parameter Set
	F - User Program

Functions

Command Class	Command Subclass
S - Function Class	A - Access Type
	H - Maximum Value
	L - Minimum Value
	N - Argument Label
	R - Argument attributes in a row format
	T - Function name
	V - Local variable attributes in row format
	Y - Type of data

I/O Registers

Command Class	Command Subclass
R - I/O Registers	B - Current State in Binary
	C - Forcing State Change
	D - Current State in Decimal
	E - Erase all Forcing Masks
	F - Forcing Mask
	S - Binary Forcing State
	X - Current State in Hexadecimal

Sequencer Data

Command Class	Command Subclass
L - Sequencer List	T: Name of Sequence list
	P: Print or Store Sequence tables
Q - Sequence Table	1: Argument 1
	2: Argument 2
	3: Argument 3
	4: Argument 4
	5: Argument 5
	F - Function number
	P - Print/Store function numbers and arguments
	T - Table Name

PLS

Command Class	Command Subclass
W-PLS	O: Phase Offset
	R: Assigned Register
	T: Master Type
	H: On & Off Value List
	E: An element's On Value
	F: An element's Off Value

Zones

Command Class	Command Subclass
Z - Zones	S: Status
	A: Upper X Coordinate
	B: Upper Y Coordinate
	C: Upper Z Coordinate
	D - Lower X Coordinate
	E - Lower Y Coordinate
	F - Lower Z Coordinate
	L - List all elements in one row

A.4 Drive and CLC / TRANS 01-D Parameters and Subclasses

The CLC / TRANS 01-D System/Axis, Task, and Drive parameters follow the same general format. The subclasses are data elements (data, name, units, etc.) as specified in SERCOS. CLC / TRANS 01-D parameters include system configuration data that is entered during the configuration of the system, as well as continuously updated system status values and messages that monitor system operation.

- CLC / TRANS 01-D card parameters are accessed using parameter set C1.
- CLC / TRANS 01-D axis parameters use the parameter sets A1 through A4.
- DDS drive parameters are accessed using parameter sets D1 through D4.
- CLC / TRANS 01-D task parameters use the parameter set Task C.

Additional information on the parameter sets can be found in the section on CLC / TRANS 01-D Parameters in this manual, and in the DDS-2 Drive and SERCOS manuals.

Example:

```
>1 CP 1.122 $cs\r\n
   || | |_ number
   || |_ set: Axis, Task, or Drive Address
   ||_ subclass: Parameter data, name Text, High or Low
limits, Attributes
   |_ class: type of parameter
```

Parameter Data Subclass

The P subclass specifies the actual parameter data, sent and received in ASCII format according to its attributes (decimal, hexadecimal, text, etc.).

Name Text Subclass

The T subclass provides the name of the parameter, provided as a text string in the language selected for the CLC / TRANS 01-D. The ability to specify actual text for the parameter name permits the host software to be independent of CLC / TRANS 01-D or Drive parameter updates.

Units Text Subclass

The U subclass returns the system units, "in" (inches) or "mm" (millimeters), as an ASCII text string.

Upper Limit, L: Lower Limit Subclasses

The H and U subclasses return the range of permissible data entry that is set by the DDS drive or CLC / TRANS 01-D for numeric data. Limits are always returned as floating point type data.

Attribute Subclass

The A subclass request a hexadecimal longword (16 bits). Bits in this longword are set for data type and scaling according to the SERCOS specification. The attribute data is available for informational purposes, or may be used to detect if a SERCOS parameter is a command or a status value.

Parameter Lists Subclasses

The D subclass is used to request lists of parameters. Since new versions of DDS drives and the CLC / TRANS 01-D may expand or change the parameter sets, lists of all parameters and all required parameters can be uploaded by the Host program. Parameters such as the DDS oscilloscope function also use variable-length lists. Parameter list formats are described later in this section.

SERCOS Parameter Sets

The SERCOS specifications allow a digital drive to have both Standard and Product Specific Parameter Sets. The Standard parameters are accessed by the parameter number (E.G., 1.95). Product Specific Parameters can be accessed using a 'P' prefix, which adds 32768 to the parameter number. For example, Parameter P-0-0005 can also be accessed as "1.32773" or "1.P5".

Example: The Host requests the name, data, and units for Drive 1 parameter 123:

```
>1 DT 1.123 \r\n  ;request drive 1 text name for
parameter 123

>1 DP 1.123 \r\n  ;request drive 1 parameter data

>1 DU 1.123 \r\n  ;request drive 1 units of measurement
```

The DDS drive responds, through the CLC / TRANS 01-D:

```
>1 DT 1.123 Feed Constant $s\r\n ;the name is "Feed
Constant"

>1 DP 1.123 6.2832 $s\r\n ;the parameter value is 6.2832

>1 DU 1.123 mm $s\r\n  ;the measurement units are in mm
```

A.5 Parameter Lists

Some CLC / TRANS 01-D functions and parameters, and SERCOS parameters, are implemented as variable-length data lists. Lists of parameters are used to determine all the parameters present on the DDS drive or CLC / TRANS 01-D card, and to classify or request parameters by function or type. The DDS-2 oscilloscope function data tables are accessed as parameter lists. Sequence numbers are used to list each parameter in the list, allowing other transmissions to the CLC / TRANS 01-D during the list upload.

List Parameter Command:

```
>1 xD a.x.n \r\n
   || | | |_ Step: Sequence number (0 to length+1)
   || | |_ Number: Parameter number
   || |_ Set: Axis, drive, or task number
   ||_ Subclass: Command, List Parameter or Table
   |_ Class: A=axis, C= card, D=drive, T=task
```

Listing a Parameter

To request a parameter list, the Host sends the list parameter command with the sequence number = 0 to the CLC / TRANS 01-D. The CLC / TRANS 01-D responds with the sequence number replace with a count of the number of items in the list. The Host then requests each list item sequentially, beginning with sequence number = 1. The sequence number is then incremented by one and the request repeated until all needed items or the entire list has been received.

The CLC / TRANS 01-D requires parameter list sequence numbers to be incremented sequentially. If an error occurs, the request for the current list item may be immediately repeated, allowing the Host to request missed data and ensuring that the data is sent in the proper order. The CLC / TRANS 01-D will respond with an error if sent an invalid sequence number.

At the end of the upload, the Host must signal the CLC / TRANS 01-D to close the list by sending a sequence number equal to the length of the list + 1.

If required, more than one parameter list can be active at one time. The Host must always close a list when it is finished since each open list uses CLC / TRANS 01-D resources.

Example of Parameter List request:

1) The Host requests a List Parameter:

```
>1 DD 1.17.0 \r\n       ;Parameter S-0-17
```

The CLC / TRANS 01-D responds with length of the list:

```
>1 DD 1.17.0 180 $s\r\n ;180 parameters in list
```

2) The Host requests the first parameter in the list:

```
>1 DD 1.17.1 \r\n
```

The CLC / TRANS 01-D responds with the first parameter:

```
>1 DD 1.17.1 44 $cs\r\n
```

3) The Host requests the second parameter in the list:

```
>1 DD 1.17.2 \r\n
```

The CLC / TRANS 01-D responds with the second parameter:

```
>1 DD 1.17.2 104 $cs\r\n
```

.
.
.

180) The Host continues to sequentially request items in list.

181) The Host closes the list by sending a sequence number = the list length+1

```
>1 DD 1.17.181 \r\n
```

The CLC / TRANS 01-D acknowledges the end of the list:

```
>1 DD 1.17.181 !19 List is finished c$s\r\n
```

Parameter List Block Transfer

Classes: C, A, D, T

Subclass: B

Data Type: List of space-delimited strings with ASCII integers or floats

For faster communications, the CLC / TRANS 01-D can send and receive parameter lists 16 elements at a time. Drive parameter lists allowing block transfer include cam tables, oscilloscope data, and any other non-text parameter list. The 'B' subclass works similar to the 'D' parameter list subclass, but instead of sending one item at a time, 16 elements are sent.

```
>u xB a.s.n \r\n
```

```
||  |  |  |_Step: Sequence number (0 to length+1)
||  |  |_  Number: Parameter number
||  |_Set: Axis, drive, or task number
||_Subclass: Command, List Parameter or Table
|_Class: A=axis, C= system, D=drive, T=task
```

Requesting a Block List Parameter

To request the start of a list, the Host sends sequence number 0 to the CLC / TRANS 01-D. The CLC / TRANS 01-D responds with the number of elements in the list.

The number of steps in the list is equal to ((elements +15)/ 16). The Host requests this number of steps from the list until the list is finished.

The data in the response strings is space delimited. Floating point and decimal values are scaled the same as when they are printed individually.

If the number of elements in the list is not evenly divisible by 16, the CLC / TRANS 01-D will fill the last response string with space-delimited zeros for each remaining element. If the data cannot be printed in less than 220 characters, the error message “!55 List or string is too long” is issued.

The CLC / TRANS 01-D requires that step numbers be incremented by one, but any previous step may be repeated. This allows the host to request any missed data and ensures that the data is sent in the proper order. For example, the sequence (1, 2, 3, 3, 4) is valid, but (1, 2, 3, 5) is not. If an invalid step number is sent, the CLC / TRANS 01-D responds with an error.

At the end of the upload, the Host must close the list by sending a sequence number equal to (length of list + 1). The Host must always close a list when it is finished since each new list uses CLC / TRANS 01-D resources.

Example:

0) Host requests a list parameter using block transfer

>1 DB 1.32840.0 \r\n ;Parameter P-0-72 (cam table 1)

CLC / TRANS 01-D responds with the number of elements in the list:

>1 DB 1.32840.0 1024 $cs\r\n ;1024 points in cam table = 64 steps

1) Host requests first 16 elements in list:

>1 DB 1.32840.1 \r\n

CLC / TRANS 01-D responds with first 16 elements:

>1 DB 1.32840.1 0.0 0.0015 0.002 0.01 0.015 --11 more elements…-- \r\n

2) Host requests elements 17-32:

>1 DB 1.32840.2 \r\n

CLC / TRANS 01-D responds with next 16 elements

>1 DB 1.32840.1 20.0 20.0015 --14 more elements…-- \r\n

3-64) Host continues to request items in list as above.

65) To close the list, host sends sequence number (steps+1)

>1 DB 1.32840.65 \r\n

CLC / TRANS 01-D acknowledges end of list:

>1 DB 1.32840.65 !19 List is finished $cs\r\n

Sending a Block List Parameter

To start sending a block list, the Host sends sequence number 0 to the CLC / TRANS 01-D, along with the number of elements to be sent. The number of steps in the list is equal to ((elements +15)/ 16). The Host sends this number of steps from the list until the list is finished.

The data in the strings must be space delimited. The host can send the data with any resolution, with or without decimal point.

If the number of elements in the list is not evenly divisible by 16, the host must fill the last string with space-delimited zeros for each remaining element.

If the number of elements in the string is less than 16, the CLC / TRANS 01-D responds with the message “!54 List or String is too short”. If the length of the data portion of the string sent to the CLC / TRANS 01-D (minus protocol header, checksum, and terminator) is greater than 220 characters, the CLC / TRANS 01-D responds with the message “!55 List or string is too long”.

At the end of the download, the Host must close the list by sending a sequence number equal to (length of list + 1). The string for this step must include at least one data element. For simplicity, the host can send 16 space-delimited zeros.

Example:

0) Host starts sending a list parameter using block transfer

>1 DB 1.32840.0 1024 $cs\r\n ;Parameter P-0-72 (cam table 1)
;1024 points in cam table = 64 steps

CLC / TRANS 01-D responds with an acknowledgment:

>1 DB 1.32840.0 $cs\r\n

1) Host sends first 16 elements in list:

>1 DB 1.32840.1 0.0 0.0015 0.002 0.01 0.015 --11 more elements…-- $cs\r\n

CLC / TRANS 01-D acknowledges:

>1 DB 1.32840.1 $cs\r\n

2-64) Host continues to send items in list as above.

65) To close the list, host sends sequence number (steps+1), with string having at least one zero.

>1 DB 1.32840.65 0.0 $cs\r\n

CLC / TRANS 01-D acknowledges end of list:

>1 DB 1.32840.65 !19 List is finished $cs\r\n

Parameter List Transfer of CLC Cams

When CLC-based cams are sent to and received from the CLC, both X and Y values are transferred. Therefore, when the block data transfer method is used, only 8 rows of the cam file are transferred per string rather than 16. Therefore, the steps in the list is equal to ((elements +7)/ 8). The data in the string is transferred as: “X1 Y1 X2 Y2 X3 Y3”, etc. If the CLC returns a count of 1025, the host must ask for 129 strings, plus the closing sequence.

Example:

0) Host requests CLC cam #1

>1 CB 1.3101.0 \r\n ;Parameter C-0-3101 (cam table 1)

>1 CB 1.3101.0 1025 \r\n ;CLC indicates 1025 points (129 steps)

1-129) Host requests data and CLC sends

>1 CB 1.3101.1 \r\n

>1 CB 1.3101.1 0.0 0.0015 0.002 0.01 --12 more …-- $cs\r\n

130) Host closes list

>1 CB 1.3101.130 \r\n

>1 CB 1.3101.130 !19 List is finished $cs\r\n

A.6 User Program Variables

The CLC / TRANS 01-D maintains a unique set of integer and floating point variables for each user program. An additional set of integer and floating point global variables is not related to a specific program and may be accessed by any program or device on the bus. User variable data can be exchanged between the Host and the CLC / TRANS 01-D using the same format as the floating point and integer parameters. The current value of a variable is obtained and changed using the P subclass.

Format:

```
>1 IP h.xx
   || | |_ number: variable table index number
   || |_ set: Program handle
   ||_ subclass: P=Send/receive Data, T=print text label
   |_ class: I=Integer Variable, F=Float Variable
            G=Global Integer,  H=Global Float
```

The user program handle provides access to the variables for any CLC / TRANS 01-D resident user program. Use the program handle "0" to access the active program's variables.

'P': Data

Type: Floating Point ("F") or Integer ("I")

Data in a CLC / TRANS 01-D variable table is accessed by supplying the class (I or F), and the numeric index (e.g. 1 for I[1] or 15 for F[15]) of the desired variable. The variable number "0" is used to request a count of the variables used by the selected program.

'T': Label Text

Type: String

The text label for any variable can be obtained by using the T subclass. If no text label is found, an ASCII space (" ") character is returned. Since the program labels are fixed when the program is compiled, labels cannot be changed with this command.

Examples:

The Host requests the number of integer variables used by program 1:

```
>1 IP 1.0 \r\n
```

The CLC / TRANS 01-D responds with:

```
>1 IP 1.0 20\r\n    ;20 variables
```

The Host sends floating point data, 123.456 to Variable F12 for the program with handle 2:

```
>1 FP 2.12 123.456 $s\r\n
```

The CLC / TRANS 01-D acknowledges with:

```
>1 FP 2.12 $cs\r\n
```

The Host requests the label name for Variable I20 for the current program:

```
>1 IP 0.20 \r\n
```

The CLC / TRANS 01-D returns the name "count":

```
>1 IP 0.20 count\r\n
```

A.7 PID

PID information is only available from the active program.

Getting the assigned variables

The host request the control register of PID loop 2 of active program.

>0 MR 0.2 $cs\r\n

The CLC responds with register number, 0 if not assigned.

>0 MR 0.2 121 $cs\r\n

The host request the status register of PID loop 2 of active program.

>0 MS 0.2 $cs\r\n

The CLC responds with register number, 0 if not assigned.

>0 MS 0.2 122 $cs\r\n

The host request the variables of PID loop 2 of active program.

>0 MB 0.2 $cs\r\n

The CLC responds with set point variable, feedback variable, output variable, and start of control block.

>0 MB 0.2 F70 F71 F72 F143 $cs\r\n

The host request the valid PID loop numbers of active program.

>0 ML 0.0 $cs\r\n

The CLC responds with loop numbers 2,3,7,8,9. Other loops are not used

>0 ML 0.0 2 3 7 8 9 $cs\r\n

The CLC responds with 0. No loops are not used

>0 ML 0.0 0 $cs\r\n

Control Blocks Variables

The control block variables are defined by the type of PID loop. The usage for type 1 is given above.

A.8 Point Tables

Each coordinated motion program stored on the CLC card has a unique set of absolute and relative points used to define the endpoints of the standard geometry segments. Each point contains a number of descriptive elements. All the elements of a point in either the Absolute or Relative Tables may be accessed individually or as a list. The available elements are shown in the list of Command Classes and Subclasses at the beginning of this section, and are described in the section on Programming Elements.

General format:

```
>1 XS h.nn \r\n
   || | |_ number: Point number
   || |_ set: Program handle
   ||_ subclass: Speed request
   |_ class: X=ABS, Y=REL
```

The program handle selects which of the user programs is accessed. To request the program points for the active program, use a "0" as the handle.

Point data is accessed by specifying a point number, that is used as a numeric index into the specified point table (E.G., 1 for ABS[1] or 21 for REL[21], the absolute or relative table is specified by the class). To request the total number of points in a table, use a "0" for the point index number.

Point Table Data

The X, Y, Z, R, P, and W coordinate subclasses are in the same format as floating point variables and parameters. The S, A, D, and J subclasses are in "xxx.x" format. Event subclasses 1, 2, 3, 4, and E are in the same format as integer variables. All subclasses have read/write access. The V subclass is a List of all user defined labels.

The following table lists each subclass, the type of units, the data type, and the data size.

Subclass	Units	Type	Size
X - X-coordinate	distance	Floating	32-bit
Y - Y-coordinate	distance	Floating	32-bit
Z - Z-coordinate	distance	Floating	32-bit
B - Blend Radius	distance	Floating	32-bit
S - Speed	% of max speed	Integer	8-bit
A - Acceleration	% of max acceleration	Integer	8-bit
D - Deceleration	% of max deceleration	Integer	8-bit
J - Jerk Limit	percent	Integer	8-bit
1 - Event 1	event number	Integer	16-bit
2 - Event 2	event number	Integer	16-bit
3 - Event 3	event number	Integer	16-bit
4 - Event 4	event number	Integer	16-bit
R - Roll R- Rate (Kinematic 8)	degrees distance/time	Floating Floating	32-bit 32-bit
P - Pitch	degrees	Floating	32-bit
W - Yaw	degrees	Floating	32-bit
E - Elbow State	none	Integer	8-bit

Examples:

```
>1 YJ 0.12 \r\n              ;host requests the jerk
limit for point REL[12]      ;in the active program.

>1 YJ 0.12 50.0 $cs\r\n      ;CLC responds

>1 X1 3.1 2 $cs\r\n          ;host selects event 2 for
                             ;ABS[1]'s first
                             ;event for program 3

>1 X1 3.1 $cs\r\n            ;CLC acknowledges
```

Point Table Data, Row Format

The data for a point is obtained in a rectangular format, similar to a row in a spreadsheet program. A regular format allows faster downloading and easier conversion to and from spreadsheets and editor programs.

As returned by the CLC, each of the 12 elements of a point are separated by an ASCII space character, and the entire point table entry is terminated by a newline. Any number of spaces between elements is allowed when uploading to the CLC.

Examples: The Host requests number of absolute points in program 2.

```
>1 XL 2.0 \r\n
```

The CLC responds with 10 points.

```
>1 XL 2.0 10 10 10 10 10 10 10 10 10 10 10\r\n
          |_ number of points
```

The Host requests point ABS[1]'s data for the active program:

```
>1 XL 0.1 \r\n
```

The CLC Responds:

```
>1 XL 0.1 1.0 2.0 0.0 0.25 50 10 33 50 1 0 0 0.0 0.0
0.0 0
;where    X   Y   Z   B    S  A  D  J  1 2 3 4 R   P
W   E
```

The Host sends point REL[2]'s data for program 3:

```
>1 YL 3.2 1.0 2.0 0.0 0.25 100 60 33 50 1 10 0
$cs\r\n
```

The CLC Acknowledges:

```
>1 YL 3.2 $cs\r\n
```

List of All User-defined Labels (V)

Label:	integer followed by string of <= 21 byte
Set:	program handle
Number:	index for list
Step:	not used

For all classes that include user-defined labels, this subclass lists the labels. To start the list and get the total number of labels, request data with Step = 0. The labels are listed in the format '*Label_num Label_string*' from the serial port. The type 'Label' is used from the executive interface. Label_num is the index in the corresponding table; for example '1' for ABS[1]. Label_string is a string of 21 bytes maximum, including null termination.

A.9 Event Tables

Each program stored on the CLC has an associated event table. Each element of the Event Table may be accessed individually or as a list. The available elements for the Event Table are shown in the list of Command Classes and Subclasses, and described in the Programming Elements section.

General Format:

```
>1 ES h.nn\r\n
   || | |_ number: Event number
   || |_ set: Program handle
   ||_ subclass: Status request
   |_ class: Event Table
```

The user program handle specifies which user program is accessed. To request events for the active program, use a "0" as the handle.

Event data is accessed by using an event number as an index into the event table (E.G. 1 for EVT[1]). To request the total number of entries in the event table, use a "0" for the event index number.

Event Table Data

Most data in the event table has associated codes corresponding to events in the teach pendant display.

Subclass	Selections	Type	Access
S Status	0= inactive 1= queued 2= pending 3= executing 4= done	Unsigned Integer	Read-only
T Type	0= undefined 1= timer based 2= coord. time 3= coord. % distance 4= single-axis distance 5= repeating axis position 6= task external interrupt input 7= VME broadcast interrupt 8= VME short address interrupt 9= axis feedback capture	Unsigned Integer	Read/write
D Direction	0= start of move 1= end of move	Unsigned Integer	Read/write

A Argument	A numeric value: (milliseconds if timed event) (% of the segment distance if coordinated motion) (degrees if repeating axis position event) Contains the probe position read from the drive, if feedback capture event is selected. ***See Programming Concepts for more information.***	Floating	Read/write
F Function	Valid event function mark	String	Read/write
M Message	text from 0-80 characters	String	Read/write

Examples:

```
>1 ES 2.0 \r\n
;Host requests number of events in program 2.
>1 ES 2.0 50\r\n
;50 events
>1 ES 1.20 \r\n
;Host req. EVT[20].s for program 1
>1 ES 1.20 1 $cs\r\n
;CLC responds with ACTIVE
>1 ET 0.10 3 $cs\r\n
;Host selects active EVT[10] as distance
>1 ET 0.10 $cs\r\n
;CLC acknowledges
>1 EF 0.1 Gripper_On $cs\r\n
;Host selects event function
>1 EF 0.1 !32 Label not found $s\r\n
;Label was not found in program
```

Event Table Data, Row Format

The data for each event is transferred between the CLC and Host in a regular format that permits easy exchange of data with spreadsheet and text editing programs. Each element of an event is separated by a single space when sent from the CLC. When sending data to the CLC any number of spaces may be used between the elements of an event. The CLC will not return an error if an event's status element is sent; the element will be ignored.

Examples: The Host requests EVT[1]'s data for program 5:

```
>1 EL 5.1 \r\n
```

The CLC responds with:

```
>1 EL 5.1 1 2 0 90.0 In_Zone In the zone! $s\r\n
          S T D A    F       M
```

The Host sends data for program 2, EVT[19]:

```
>1 EL 2.19 0 1 1 1000 Calc_Zone Calculating... $s\r\n
```

A.10 CLC Program Communication

Each user program (text or icon) that is developed and successfully compiled on a PC Host results in a downloadable base-code CLC executable file. These executable files may be exchanged between the Host and CLC using the upload/download method described in this section. Programs resident on the CLC can be activated and deleted; and lists of variables and subroutines associated with the program may be accessed using the serial communication interface. The number of programs that may be downloaded to a CLC is limited by the size of the programs and the amount of available non-volatile memory.

Program Header Record

Each user program contains a header record that identifies the program and is used for starting program uploads and downloads. The header contains the program name, size, checksum, and date; and is stored with the program in the CLC's non-volatile memory.

The size is a decimal count of all bytes in the program file. The checksum is a 4 byte hexadecimal two's complement sum of all bytes in the file (excluding the checksum). The date is in Month/Day/Year format (E.G., 07/04/93). The program name can include any ASCII characters and may be of any length. The detailed CLC program file format is described in an appendix.

Download Block Size

CLC program file uploads and downloads are performed as a sequence of fixed-length blocks. The block size is selected using Parameter C1.50 (Download Block Size) and defaults to 16 bytes. The Download Block Size parameter may be changed to 1 to 512 bytes by the Host computer to optimize transmission time or for terminal compatibility.

Initializing a Download (PW)

To start a download from the Host to the CLC, the Host sends a Header Record with the "PW" command. If the available CLC memory is insufficient to contain the program, the CLC returns an error message. If enough memory is available, the CLC responds with a numeric program handle. The program handle returned by the CLC must be used to identify the program with subsequent commands. The CLC will always respond with a new handle, allowing more than one program with the same name.

Example: The Host starts a download:

```
          |<--------Header Record------->|
>1 PW 0.0 1592 ABCD1234 12/25/92 Program_1 c$\r\n
    |     |    |        |        |_ Program Name
(variable length)
    |     |    |        |_ Date (mm/dd/yy)
    |     |    |_ Program checksum (hex)
    |     |_ Program size in bytes (decimal)
    |_ Command: Start Download
```

The CLC responds with a valid program handle:

```
>1 PW 0.0 1 $cs\r\n
          |_ Program handle returned by CLC
```

Executing a Download (PD)

After the Host has sent the program header record and obtained a program handle, the Host must send the entire program to the CLC, before the program may be activated. The Host uses a sequential number to identify each block. Block by block transfer permits other communication with the CLC to take place between blocks.

If a communication error is received when sending a program block, the current block may be repeated. The CLC download program requires program block sequence numbers to be incremented by one, or an error message will be returned.

After all program blocks have been downloaded, the Host completes the download by sending a final block with the data area containing at least one non-space character. The final block is identified by a sequence number equal to the last program block sequence number + 1. The checksum is verified and the CLC responds with an acknowledgment or an error message.

Note: The PD command must not be executed without a preceding PW or PR command to initialize the transfer.

Example: The Host sends a block (using the default block size= 16 bytes):

```
>1 PD 1.1 0123456789ABCDEF0123456789ABCDEF$cs\r\n
    | | | |_ 32 hex digits of data (16 bytes)
    | | |_ Sequence number (from 1 to (size/block size))
    | |_ Program handle previously obtained by PW or PR command
    |_ Command: Download
```

The CLC responds during download (no error):

```
>1 PD 1.1 $cs\r\n
```

The Host sends the final block after 100 program blocks have been sent :

```
>1 PD 1.101 0 $cs\r\n
        |   |_ dummy block not stored by CLC
        |_ (size of program/block size) +1
```

The CLC responds after verifying the checksum:

```
>1 PD 1.101 !19 List is finished $cs\r\n
```

Initializing an Upload (PR)

To request an upload from the CLC, the Host sends a program handle with the "PR" command. If the program handle does not match a program in the CLC, the CLC responds with an error. The CLC responds with the corresponding program header. (The "PH" command can be used to obtain a list all CLC resident programs and handles.)

The Host requests an upload:

```
>1 PR 0.0 1 $cs\r\n
    |     |_ Program handle
    |_ Command: Start Upload
```

The CLC responds with the program's header:

```
>1 PR 0.0 1 1592 ABCD1234 12/25/92 Program_1 c$\r\n
          | |<------- Program Header ------>|
          |_ Program handle
```

Executing an Upload (PD)

The PD command is also used during uploads, but the Host sends only the sequence number. The CLC responds with the data stored in each block. The process repeats sequentially, block by block, in the same manner as the program download (see Executing a Download, above).

The Host requests a block:

```
>1 PD 1.1 $cs\r\n
    | | |_ program block sequence number
    | |_ program handle
    |_ Command: upload
```

The CLC responds with data for the requested program and block:

```
>1 PD 1.1 0123456789ABCDEF0123456789ABCDEF$cs\r\n
```

The Host sends the dummy block request, after the last program block has been received:

```
>1 PD 1.101 $cs\r\n
        |_ (size of program/block size) +1
```

The CLC responds after checksum verification:

```
>1 PD 1.101 !19 List isfinished$cs\r\n
```

Erasing (Deleting) a Program (PE)

This command erases the program identified with the specified program handle. The CLC memory previously required by the program becomes available to other programs; and the header and handle are removed from the CLC's list of resident programs.

Notes: 1) A currently active program may not be erased, unless the CLC has been placed in Parameter Mode.

2) The Erase Program command format specifies "0" in the positions normally occupied by the program handle and block sequence number, followed by the numeric program handle of the program to delete.

The Host sends an Erase Program Command:

```
>1 PE 0.0 1 $cs\r\n
          |_ Erase Program with handle #1
```

The CLC acknowledges with:

```
>1 PE 0.0 $cs\r\n
```

List CLC Resident Programs (PH)

The "PH" command provides a list of the programs resident on the CLC. The list contains the headers of all programs and their corresponding program handles. An upload sequence of commands are used to obtain the list.

The Host requests the start of the program list:

```
>1 PH 0.0 $cs\r\n
    |   |_ sequence 0= start of list
    |_ Command: list program headers
```

The CLC responds with the number of programs on the card:

```
>1 PH 0.0 4 $cs\r\n
          |_ number of resident programs
```

Host Requests a file record from the list:

```
>1 PH 0.1 $cs\r\n
        |_ Sequence number for record 1
```

CLC Responds:

```
>1 PH 0.1 1 1592 ABCD1234 12-25-1992 Program_1 c$\r\n
        | | |<------ Program header ------->|
        | |_ Program handle for following program
header
        |_ Sequence number for program header list
```

Activating a Program (PA)

The Host can activate a program by sending its program handle with the "PA" command. If the handle or the program is invalid, the CLC returns an error message.

Note: A program can not be activated by the Host if the CLC is currently running any user tasks.

The Host sends an activate program 2 command:

```
>1 PA 0.0 2$cs\r\n
          |_ Activate program with handle = 2
```

The CLC acknowledges with:

```
>1 PA 0.0 $cs\r\n
```

Request Currently Active Program (PA)

If the Host sends a "PA" command without specifying a program, the CLC returns the handle of the currently active program. Other information about the program is available through task parameter requests.

The Host sends:

```
>1 PA 0.0 \r\n
```

The CLC returns:

```
>1 PA 0.0 2 $cs\r\n
          |_ Program 2 is active
```

Request Name of Program (PN)

The "PN" command may be used to obtain the ASCII text name of a specified program handle from the CLC. If a handle of 0 is requested, the name of the currently active program is sent. If an invalid handle is requested, the CLC returns an error message.

The Host requests the active program:

```
>1 PN 0.0 \r\n
      |_ Request active program's name
```

The CLC responds with:

```
>1 PN 0.0 prog_one\r\n
```

The Host requests the name of program 3:

```
>1 PN 3.0 \r\n
      |_ Request program 3's name
```

The CLC returns:

```
>1 PN 3.0 prog_3\r\n
```

List of Event Function Marks (PF)

Text marks for CLC resident event functions can be listed by using a list sequence. Subroutine marks are listed in alphabetical order, as stored in the CLC program file. The program handle is used to identify which program's events are required.

Request Format:

```
>1 PF h.s
      | |_ sequence number
      |_ program handle
```

Response Format:

```
>1 PF h.s Sub_Mark
          |_ subroutine mark (or function name)
```

Example: The Host requests the function mark list for the active program:

```
>1 PF 0.0 \r\n
```

The CLC responds with the count of marks for the active program:

```
>1 PF 0.0 2 $cs\r\n
          |_ 2 marks are defined
```

The Host requests the first mark:

```
>1 PF 0.1 \r\n
```

The CLC returns:

```
>1 PF 0.1 Close_Gripper $cs\r\n
```

The Host requests the next mark (2):

```
>1 PF 0.2 \r\n
```

The CLC responds:

```
>1 PF 0.2 Open_Gripper $cs\r\n
```

The Host closes the list:

```
>1 PF 0.3 \r\n
        |_ next sequence number after count of defined
marks
```

The CLC acknowledges and closes the list:

```
>1 PF 0.3 !19 List is finished $cs\r\n
```

List Variable Labels (PV)

Text labels for variables are stored with each program resident in the CLC, and can be listed using a list sequence. The variable numbers and text names are listed in alphabetical order by text name. The variable number consists of an "I" or "F" ASCII character, followed by a 1-to-3 digit number. The variable number and its label are separated with a space.

Request Format:

```
>1 PV h.s
      | |_ sequence number
      |_ program handle
```

Response Format:

```
>1 PV h.s Ixx Label
          ||  |_ variable label
          ||_ variable id number (index number)
          |_ variable type identifier (F = floating
point, I = integer)
```

Example: The Host requests the variables label list for program 1:

```
>1 PV 1.0 \r\n
```

The CLC responds with the count of named variables for program 1:

```
>1 PV 1.0 2 $cs\r\n
          |_ 2 labels are defined
```

Host requests the first label:

```
>1 PV 1.1 \r\n
```

The CLC responds with:

```
>1 PV 1.1 F11 axis1_velocity $s\r\n
```

The Host requests the next label

```
>1 PV 1.2 \r\n
```

The CLC responds with:

```
>1 PV 1.2 I1 part_count $cs\r\n
```

Host closes the list:

```
>1 PV 1.3 \r\n
```

The CLC closes the list and acknowledges:

```
>1 PV 1.3 !19 List is finished $s\r\n
```

Transfer Tables Between Programs (PX)

The CLC stores a unique set of tables for each user program. When a new program is downloaded, the new programs tables are set to default or new values. The "PX" command allows the Host to copy a working program's tables to another program. The ability to copy a set of working, tested tables to another program under test can be exceedingly useful when debugging a program, or entering/modifying points using the teach pendant.

The transfer command copies all the tables associated with the CLC resident source program to the resident destination program. The tables include: the float and integer variables, the absolute and relative point tables, and the event table.

Tables are copied on a one-for-one, table-to-table basis; with the source table overwriting the destination table. If the source and destination tables are of different sizes, the table will be copied so that the destination table size does not increase. If the source table size is less than the destination, the remainder of the destination table will keep the same values.

The event functions in the source program are checked against those that exist in the destination program and new function offsets are stored. If an event function does not exist in the destination program it is set to "NONE."

Format:

```
>1 PX s.0 d \r\n
      |   |_ program handle of destination program
      |_ program handle of source program
```

Example: The Host requests that program 3 load a copy of program 1's tables:

```
>1 PX 1.0 3 \r\n
```

The CLC responds with:

```
>1 PX 1.0 \r\n
```

Example: The Host requests that program 1 load a copy of the active program's tables:

```
>1 PX 0.0 1 \r\n
```

The CLC responds with:

```
>1 PX 0.0 \r\n
```

CAM Indexer function block variables (PJ)

Class P: Program

Subclass J: Registration Block

This communications request allows a user interface to determine the variables assigned to a CAM Indexer function block. A space delimited string in the following format is returned from the CLC: “cam_id Fstart_float Istart_int”. The CAM Indexer information is avaliable only from the active program.

Cam_id	Cam number, range 1 to 8
start_float	Starting variable of the float data block
start_int	Starting variable of the integer data block

In the data request, the Set (before the decimal point) is always equal to 0, indicating the active program. The Number (after the decimal point) is the cam id number block number. When block 0 is requested, the total number of active CAM Indexer blocks is returned.

Example

Host: Ask for number of CAM Indexer blocks active in the active program

>0 PJ 0.0

CLC: No CAM Indexer blocks are present

>0 PJ 0.0 0

CLC: Two CAM Indexer blocks are present

>0 PJ 0.0 2

Host: Ask for CAM Indexer block 1 information

>0 PJ 0.1

CLC: cam 1, floats at F10, ints at I10

>0 PJ 0.1 1 F10 I10

Sequencer/Subroutine Related Subclasses

Subclasses:	L Max number of rows in sequence list (integer, read only) Q Max number of rows in sequence table (integer, read only) S Number of function slots available (integer, read only) I Data change identification (integer, read only)
Set:	program handle
Number:	for S subclass: 1=total number, 2=current available for Q, L subclasses, always set to 0 for I subclass, 1=sequencer system
Step:	not used
Exec functions:	get_Program(), put_Program()

These classes display the limits on sequence lists and tables that were set up at compile time.

The **L** subclass displays the maximum number of rows allowed in a sequence list.

The **Q** subclass displays the maximum number of rows allowed in a sequence table. This number is internal to the CLC and is set to 100 in the current version.

The **S** subclass displays information about the memory used for sequence tables. The functions entered into a sequence table are stored in memory as an array of function slots that is used by all sequences, which optimizes memory usage and editing. If ‘number’ is set to 1, the total number of function slots is returned. If ‘number’ is set to 2, the number of unused slots for functions remaining is returned.

The **I** subclass provides a way for a user interface to detect if data has been changed by another port or the user program. Each time data is stored, a counter is incremented. To check the sequencer system, set ‘number’ to 1. Other numbers are reserved for future functions.

Registration Block Information

Class P: Program

Subclass M: Registration Block

This communications request allows a user interface to determine the variables and registers that are assigned to a registration function block. A space delimited string in the following format is returned from the CLC: “probe_axis Fstart_float Istart_int Rcontrol_reg Rstatus_reg”. The registration information is avaliable only from the active program.

probe_axis	Axis (1-40) with the probe measurement value
start_float	Starting variable of the float data block
start_int	Starting variable of the integer data block
control_reg	Registration control register
status_reg	Registration status register

In the data request, the Set (before the decimal point) is always equal to 0, indicating the active program. The Number (after the decimal point) is the registration block number. When block 0 is requested, the total number of active registration blocks is returned.

Example

Host: Ask for number of registration blocks active in the active program

>0 PM 0.0

CLC: No registration blocks are present

>0 PM 0.0 0

CLC: Two registration blocks are present

>0 PM 0.0 2

Host: Ask for registration block 1 information

>0 PM 0.1

CLC: axis 1, floats at F10, ints at I10, control at R101, status at R102

>0 PM 0.1 1 F10 I10 R101 R102

A.11 Function Class (S)

Subclasses

T	Function name	string, <= 21 byte (read only)
A	Access type	integer (1=not accessible, 2=accessible)
N	Argument label	string, <= 21 byte (read only)
Y	Type of data	1-character string: ‘I’, ‘F’, ‘X’, ‘Y’, or ‘R’ (read only)
L	Minimum value	float (read only)
H	Maximum value	float (read only)
R	Argument attributes in row format	string <= 81 byte (read only)
V	Local variable attributes in row format	string <= 81 byte (read only)

Set:	program handle
Number:	function number (0 = total number of functions)
Step:	argument number 1-5 (0= number of args for this function)
Exec functions:	get_Function(), put_Function()

All subroutines, tasks, and functions in a program have attributes that can be read using this class. These are used for displaying names and validating data. All attributes are set in the user program and cannot be changed on-line. The function number is used in all references to this function in the sequence table.

The **T** subclass returns the function name as a null-terminated string of 20 characters or less.

The **A** subclass displays (2) if the function can be used in a sequence table, (1) if it is hidden from the user.

The **N** subclass displays an argument name as a null-terminated string of 20 characters or less.

The data type **Y** is an ASCII character corresponding to the data class. This class can be used to access label lists for points and registers.

The minimum and maximum values are returned in the **L** and **H** subclasses. The user interface can read these values to determine if data entry is valid.

The **R** subclass provides the serial port interface with subclasses N, Y, L, and H for function arguments in a space delimited string. For example: “>0 SR 0.1.1 test_var F 0.0 10000.0 $cs\r\n”.

The **V** subclass provides the same information as the **R** subclass, but with local variables. Since there are no argument ranges for local variables, zeros are printed in the last two fields. The number of local variables in a task is obtained with the command “>n SV h.f.0”, where n=unit, h=handle, and f=function.

A.12 Input/Output Registers

The Host system may read the CLC / TRANS 01-D's input and output registers at any time; including control, status, and programmable registers. The CLC / TRANS 01-D's axis, system, and task status registers are normally read-only, and are only changed by the CLC / TRANS 01-D I/O mapper executive task or the register forcing commands (see below). Setting I/O registers directly (using the RB, RX and RD commands) has the lowest priority of all I/O access methods.

Directly accessing I/O registers should be done with caution. The CLC / TRANS 01-D is a multitasking system, and as such the potential for I/O contention always exists between user tasks, the Host communication, the I/O Mapper, and the I/O subsystem. It is the programmer's responsibility to anticipate contention problems and synchronize accesses to data between asynchronous CLC / TRANS 01-D tasks when necessary.

The forcing commands (RM, RF, RC and RS) are provided primarily for debugging purposes. Forcing commands should be used with extreme caution since they can be used to override the state of system control registers, and have higher priority than the CLC / TRANS 01-D's I/O mapper or Host direct access commands.

The requirement for a checksum may be disabled by parameter. This

practice is not suggested. It results in no communication error checking of data sent to the CLC / TRANS 01-D that may effect safe operation of the system.

I/O Register Access (RB), (RX), (RD)

Input registers are accessed using "R(data type)" commands and a register specifying index number within the range 1 to 200. The current contents of the register may be read as a 16-bit binary number (command "RB"), a 4-digit hex number (command "RX"), or a decimal integer number (command "RD").

Example:

```
>1 Rt 0.nnn $cs\r\n
    | | |_ register number
    | |_ set ID, always 0 for I/O registers
    |_ subclass: type or format B(=binary, D==decimal,
X=hexadecimal)
```

I/O Register Read

Example: Host requests the contents of register 1 in binary:

```
>1 RB 0.1 \r\n
```

The CLC / TRANS 01-D responds with:

```
>1 RB 0.1 0001001000110010 $s\r\n
          |              |_ least significant bit
          |_ most significant bit
```

The checksum is optional when reading data from the CLC / TRANS 01-D.

Sending a "0" as the register index number returns the number of registers in the current system.

I/O Register Write

The Host may send a value to a CLC / TRANS 01-D I/O register in hexadecimal ("RX"), binary ("RB"), or decimal ("RD") using the same format as an I/O read with the addition of a data field and checksum.

Example:

```
>1 RX 0.121 0x0040 $s\r\n
    | | |   |_ 16 bit hex word to write
    | | |_ I/O register number 121
    | |_ always 0 for I/O registers
    |_ read/write to register in hex
```

Set Current I/O State with Mask (RM)

The RB, RD and RX commands affect every bit of the destination I/O register, the new data word replaces the old word. RM allows you to specify a mask in addition to data bits, limiting the I/O register bits that are changed.

The most significant 16 bits in this 32-bit word provide the mask selecting the bits that may be changed. A "1" enables change, "0" masks any change. The least significant 16 bits changes the state of the I/O register bits. If RM is used to read the register, the CLC / TRANS 01-D returns the state of all bits. See the notes above for changing I/O register bits.

Example: Format:

```
>1 RM 0.2 0x00600040
            |  ||__|_ 16 bit word of new bit states
            |__|_ 16 bit mask of bits to change

>1 RM 0.2 0x00600040 ;bit 6 is turned on, bit 7 off
```

RM is a single use, independent equivalent of setting a mask with an RF command, then setting the actual I/O bit states with an RC or RS command. Since RM contains its own mask it does not affect the forcing mask set with RF. See the RF, RC and RS commands below.

I/O Forcing Selection (RF)

The forcing selection (RF) and forcing state (RC and RS) commands allow the Host to selectively force the state of individual bits in the I/O registers. Forcing commands take priority over the CLC / TRANS 01-D I/O mapper and I/O devices.

The forcing remains in effect until the mask for each forced bit is cleared, or until there is a timeout error on the serial port. When the forcing state changes for bits in a CLC / TRANS 01-D control register, all edge detection is reset.

If a bit in the 16-bit forcing mask is set to 0, the corresponding bit in the I/O register is controlled by the I/O Mapper and physical I/O. If the forcing mask bit is set to 1, the I/O register bit is forced by the Host "RC" or "RS" command.

The data format of the "RF" 16-bit forcing mask word is always binary.

Example:

```
>1 RF 0.2 0000000001001000
                    |__|_ bits 4 and 7 are forced bits
and are
                         controlled by the Host all
other bits
                         are controlled by the
physical I/O and
                         the CLC / TRANS 01-D I/O
Mapper
```

I/O Forcing State Change (RC)

The most significant 16 bits in this 32-bit word select which bits in the I/O register may be affected, and the least significant 16 bits changes the states of those bits. If it is read, it returns the state of all bits.

The data format of the "RC" state change word is always a 32-bit hexadecimal longword.

Example: Format:

```
>1 RC 0.2 0x00600040
            |  ||__|_ 16 bit word of new bit states
            |__|_ 16 bit mask of bits to change

>1 RC 0.2 0x00600040
              |   |_ bit 6 on, bit 7 off
              |_ allow changes to bits 6 and 7
```

I/O Binary Forcing State (RS)

The "RS" command is used to read and write the state of the forcing bits for the selected register. If bits are to be affected, the desired bits in the I/O register must have had forcing enabled by a forcing mask set with the "RF" command.

The data format of the "RS" 16-bit forcing state word is always binary.

Example:

```
>1 RS 0.2 0000000100000001
                  |_______|_ bits 1 and 9 are turned
on, all other
                             bits turned off if the
bits have
                             forcing enabled by an "RF"
command
```

Erase All Forcing Masks (RE)

This command sets all forcing masks and states to zero and returns the I/O system to normal control. The command only takes effect at the time that it is sent.

Caution should be used when using this command. The I/O registers are directly affected and clearing the mask(s) may cause immediate unwanted motion in the system.

Example: The data format of the "RE" command is ASCII integer.

```
>1 RE 0.1 1
        | |_ set to 1 to erase forcing masks
        |_ always '0.1'
```

A.13 Sequencer Data

The CLC serial protocol includes the following classes and subclasses to handle the sequencer and functions with arguments.

Sequence List Class (L)

Subclasses

T	Name of Sequence List	string, <= 21 byte (read/write)
P	Print/Store Sequence Tables (rows of list)	integer sequence table number

Set:	program handle
Number:	Sequence List Number (0 returns total number of lists)
Step:	sequence table row in this list (0 returns number of rows)
Exec functions:	get_SequenceList(), put_SequenceList()

The **T** subclass reads or writes the sequence list name as a null-terminated string of 20 characters or less.

The **P** subclass prints or stores a sequence list as a variable-length list. The tables in the list are identified by the 'number' in the 'Q' data class.

To store a new sequence list, the host sets the 'step' to 0 and sends the size of the list. Setting the size of the list to 0 erases all entries in the list.

To change an existing row in the sequence list, 'step' must be set to the row number.

If a sequence list is edited while it is running, the communication error 'Cannot edit sequence table while running' is issued.

Sending a new sequence list:

```
>0 LP 0.1.0 5                    ;Host tells CLC to store list 1 with 5 function tables
>0 LP 0.1.0                      ;CLC acknowledges
>0 LP 0.1.1 2                    ;Host selects Table number 2 as first table
>0 LP 0.1.1                      ;CLC acknowledges
>0 LP 0.1.2 1                    ;Host selects Table number 1 as second table
>0 LP 0.1.2                      ;CLC acknowledges

>0 LP 0.1.6 0                    ;Host closes the list
>0 LP 0.1.6 !19 List finished    ;CLC acknowledges, table now ready to execute
```

Changing one row of sequence list:

```
>0 LP 0.1.2 2          ;Host changes second row in list to Table number 1
>0 LP 0.1.2            ;CLC acknowledges
```

Sequence Table Class (Q)

Subclasses

T	Table Name	string, <= 21 byte (read/write)
P	Print/Store function numbers and arguments (rows of table)	string, <= 255 byte or Special structure (read/write)
F	Function number	integer (read only)
1	Argument 1	float (read/write)
2	Argument 2	float (read/write)
3	Argument 3	float (read/write)
4	Argument 4	float (read/write)
5	Argument 5	float (read/write)

Set:	program handle
Number:	table number (0 returns total number of tables)
Step:	function row in this table (0 returns number of rows in this table)
Exec functions:	get_SequenceTable(), put_SequenceTable()

The T subclass reads and writes the sequence table name as a null-terminated string of 20 characters or less.

The P subclass prints or stores a sequence table as a variable-length list. The functions in the list are identified by the ‘number’ in the ‘S’ data class.

To store a new sequence table, the host sets the ‘step’ to 0 and sends the size of the table. Setting the size of the table to 0 erases all entries in the list.

To change an existing row in the sequence table, ‘step’ must be set to the row number.

Through the serial protocol the data is a space-delimited string in the format (function number, arg1, .. argn). Arguments are printed as floating point values. Arguments that don’t exist are printed as ‘0’. The CLC will ignore extraneous arguments when data is sent to it.

From the executive interface, the data is type Special with the structure FUNC_DATA_t. All elements of a function are passed and returned using this structure (see end of this document).

The function name can be requested with the “ST” protocol command. Argument names, types, and limits can be requested with the ‘S’ data class.

If a sequence table is edited while it is running, the communication error ‘Cannot edit sequence table while running’ is issued.

Subclasses F, 1, 2, 3, 4, and 5 can be used to individually access elements of each row of the table. The individual arguments can be changed while the sequence table is running.

(refer to example for sequence list for sending data)

Getting a sequence table from the CLC:

```
>0 QP 0.2.0                          ;Host asks for number of functions in table 2
>0 QP 0.2.0 3                        ;CLC responds
>0 QP 0.2.1                          ;Host asks for first function
>0 QP 0.2.1 2 12.34 100 25 0 0       ;function number 2
>0 QP 0.2.2                          ;Host asks for second function
>0 QP 0.2.2 5 100 0 0 0 0            ;function number 5
>0 QP 0.2.3                          ;Host asks for second function
>0 QP 0.2.3 1 12.5 30.2 0 0 0        ;function number 1
>0 QP 0.2.4                          ;Host closes the list
>0 QP 0.2.4 !19 List is finished     ;CLC acknowledges
```

A.14 PLS

Class	Subclass
W: PLS(Programmable Limit Switch)	**A**: Master Axis Number
	O: Phase Offset
	R: Assigned Register
	T: Master Type
	H: On & Off Value List
	E: An element's On Value
	F: An element's Off Value

Format:

>1 Wn h.n\r\n

| | | |_number: Switch number

| | |_set: Program handle

| |_subclass: n = A, N, O, R, T

|_class: PLS

The number of switches per program can be requested by sending any request with number equal to 0.

Examples:

Host requests PLS count for program 5:

>1 WR 5.0 \r\n

CLC responds one PLS:

>1 WR 5.0 1

Host requests assigned register to PLS 1 of program 4:

>1 WR 4.1 \r\n

CLC responds register 100:

>1 WN 4.1 100

Format of Configuration Data

A Master Axis	Axis Number(1 - 40)	Integer	8-bit
R Assigned Register	0=Not Active, 1-1024	Integer	8-bit
O Offset from Master	distance/degrees	Floating	32-bit
T Master Type	1=ELS, 2=VM, 3=AP, 4=AS	Integer	8-bit

>1 WH h.n.m\r\n
|| | | |_element
|| | |_number: Switch number
|| |_set: Program handle
||_subclass: Switch Values
|_class: PLS

The number of elements per switch can be requested by sending a request with element equal to 0.

Format of Switch Values

H Switch Values	On & Off Values	Floating	32-bit

Example:

Host requests element count of PLS 1 for program 5:
>1 WH 5.1.0 \r\n
CLC responds 16 elements:
>1 WH 5.1.0 16
Host requests on & off values for PLS 1, element 6, of program 4:
>1 WH 4.1.6 \r\n
CLC responds:
>1 WH 4.1.6 98.0 167.9
Host requests the on value for PLS 1, element 6, of program 4:
>1 WE 4.1.6 \r\n
CLC responds:
>1 WE 4.1.6 98.0
Host requests the off value for PLS 1, element 6, of program 4:
>1 WF 4.1.6 \r\n
CLC responds:
>1 WF 4.1.6 167.9

Drive PLS ASCII Protocol

Some SERCOS drives have a PLS associated with them (current versions allow only one). Elements of the PLS can be accessed individually or as a list. The available elements are described in the PLS Help in Visual Motion.

List

Host requests the count of On element's for drive 1:

>1 DD 1.32900.0 \r\n

CLC responds:

>1 DD 1.32900.0 8

Host requests the first On element for drive 1:

>1 DD 1.32900.1 \r\n

CLC responds:

>1 DD 1.32900.1 25

Host requests the last On element for drive 1:

>1 DD 1.32900.8 \r\n

CLC responds:

>1 DD 1.32900.8 350

Host requests one more element to close list for drive 1:

>1 DD 1.32900.9 \r\n

CLC responds:

>1 DD 1.32900.9 !19 List is finished

Individual elements

The subclass 'E' of the ASCII protocol for drive parameters allows individual elements of the list to be accessed.

Host requests the 5th Off element for drive 1:

>1 DE 1.32901.5 \r\n

CLC responds:

>1 DE 1.32901.5 270

Host writes the 4th Offset element for drive 3:

>1 DE 3.32902.4 220\r\n

CLC responds:

>1 DE 3.32902.4

B DDE Client Interfaces

The following examples illustrate how to create custom DDE client interfaces for the CLC. DDE communication allows certain software applications to read from (*request*), and write to (*poke*) the CLC card. See the *CLC Reference Manual, Appendix A. CLC DDE SERVER* for more information.

B.1 Creating and Customizing a DDE Client Interface in Microsoft Excel™

The following example illustrates how to create a custom DDE client interface for the CLC card using Microsoft Excel™ (*Version 5.0 and up*) Other programs that support DDE communication can also be used in a similar fashion. Requested information can be read directly by a spreadsheet, chart or database, while poke transactions allow users to control program execution from within this custom interface.

The example below was created for an ELS/CAM application. The instructions for creating the CLC program that corresponds with this spreadsheet can be found in the *Appendix E. Example Programs*. All of the *requested* data in the spreadsheet was read from the CLC card with the CAM program running.

DDE Worksheet Functions

A DDE request can be made directly from a *cell* within an Excel™ Worksheet using a formula outlined below. The CLC_DDE Server should be running before a request is made. Each request is queued in the server and then handled using round robin arbitration. The Excel Worksheet will automatically update the cell as the information becomes available. The response time varies according to how many other applications are running and how many DDE conversations are occurring at the same time. Limit the number of active DDE requests within a worksheet in order to get a faster response time.

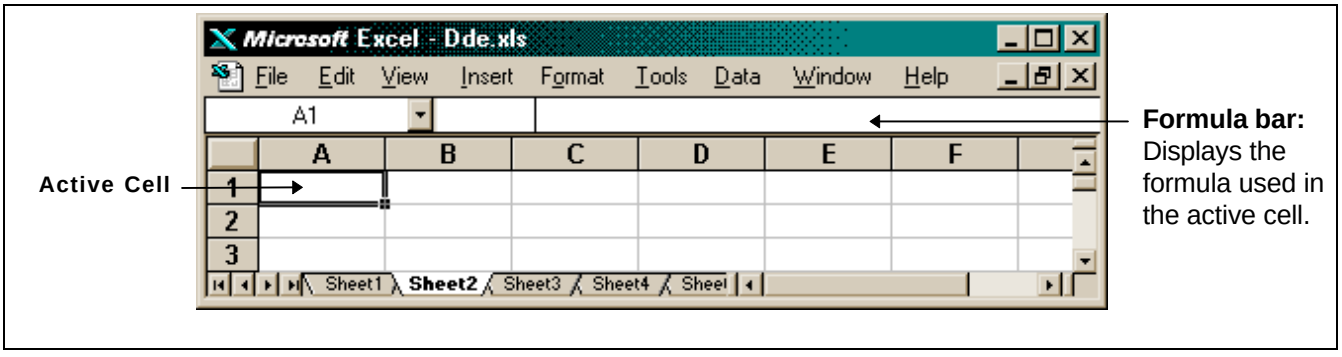
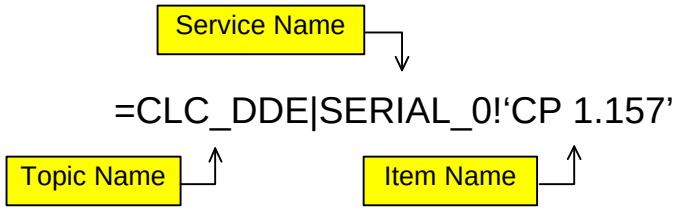

Microsoft Excel - Dde.xls
File Edit View Insert Format Tools Data Window Help
A1
A B C D E F
1 2 3
Sheet1 Sheet2 Sheet3 Sheet4 Sheet
Active Cell
Formula bar:
Displays the formula used in the active cell.

Figure B-1: Excel Worksheet

Select a cell and then enter the **DDE Service name**, **Topic name** and **Item name** within the formula bar using the following syntax as an example:



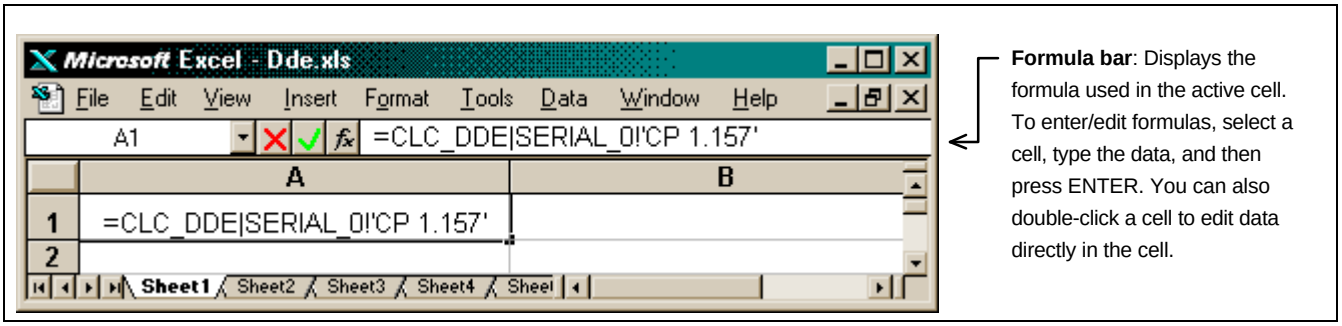

Microsoft Excel - Dde.xls
File Edit View Insert Format Tools Data Window Help
A1
=CLC_DDE|SERIAL_0!'CP 1.157'
A
B
1
=CLC_DDE|SERIAL_0!'CP 1.157'
2
Sheet1 Sheet2 Sheet3 Sheet4 Sheet
Formula bar: Displays the formula used in the active cell. To enter/edit formulas, select a cell, type the data, and then press ENTER. You can also double-click a cell to edit data directly in the cell.

Figure B-2: Example DDE Formula

The **Item name** ‘CP 1.157’ will read the value of the system or “Card” Parameter C-0-0157 which is the current ELS master position.

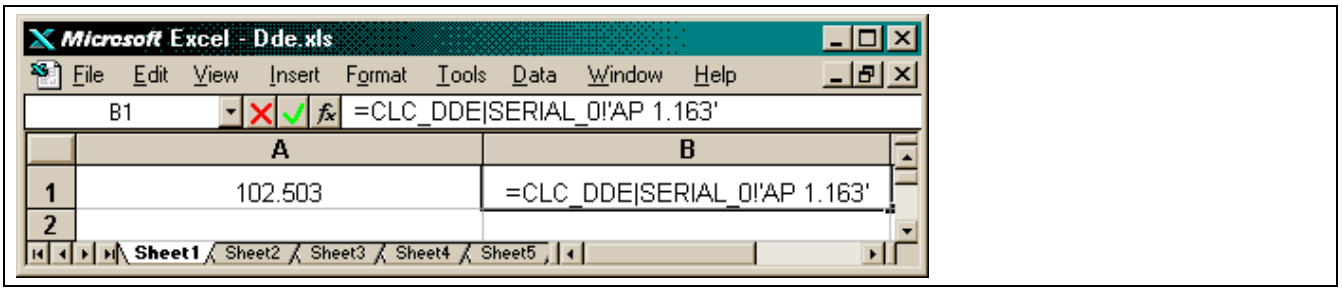
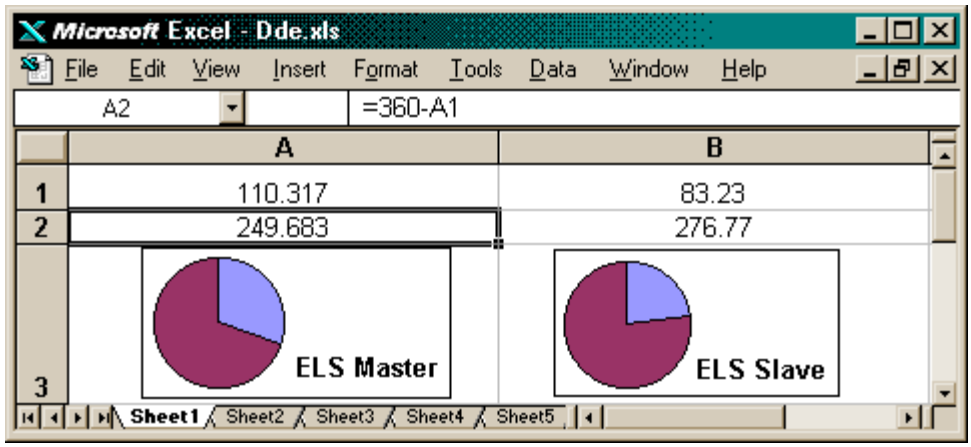


Figure B-3: Example DDE Formula Result

The **Item name** ‘AP 1.163’ will read the value of the Axis Parameter A-0-0163 which is the CLC Cam Output Position. **See *Appendix B. Direct ASCII Communications* for the syntax used with other item names.**

A pie chart can be used to graphically illustrate the master and slave positions between 0° and 360°. Enter two formulas in the second row which subtract the ELS Master and Slave Positions from 360° (=360-A1,=360-B1). Select each column and create a pie chart using the Excel *Chart Wizard*. If the CLC_DDE Server is active and an ELS program is running the pie charts will rotate to reflect the current ELS positions as they change.

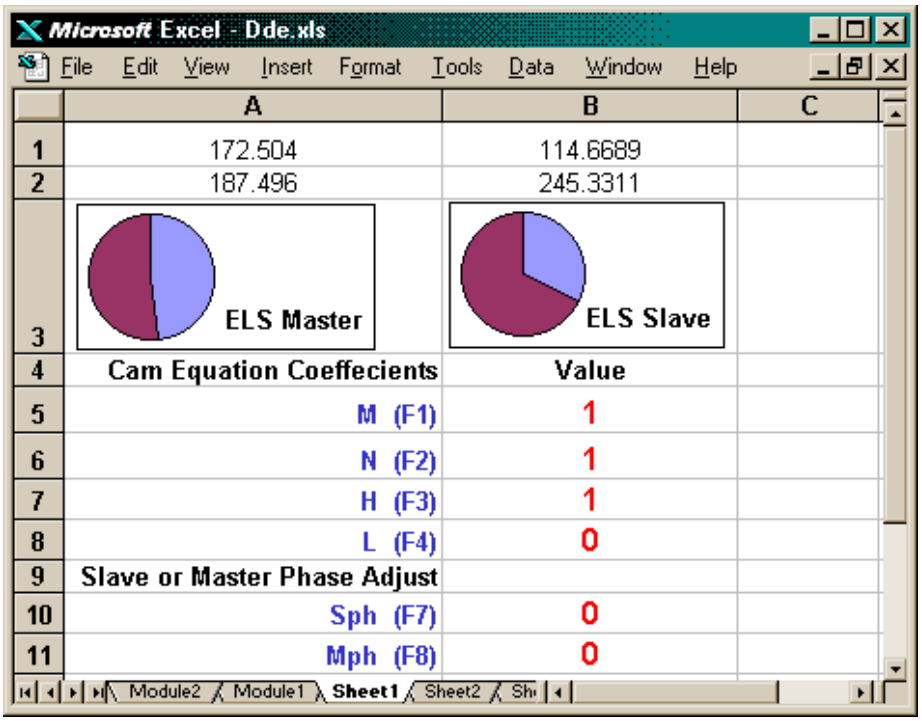


DDE Functions using Visual Basic® for Excel™

Visual Basic® for Excel™ has its own DDE Functions which can be used in a spreadsheet macro or module. The following Visual Basic® macros illustrate how to use the *DDERequest* and *DDEPoke* functions. The *DDERequest* function is used to read the values of the CAM coefficient and phase offset parameters. The *DDEPoke* function is used to write values to predefined program variables which were added in the spreadsheet. Each variable also has a corresponding **Item name** needed for DDE communication.

The variables listed in column A were predefined in the CAM program to store the CAM coefficients and Phase Adjust values. **Macro 1** requests the current coefficient values from the corresponding CLC System (card) and Axis parameters.

Note: *When making a DDERequest from a macro the Service name and Topic name are included in the DDEInitiate function and assigned to a variable (MYCLC***).**


Microsoft Excel - Dde.xls
File Edit View Insert Format Tools Data Window Help
A B C
1 172.504 114.6689
2 187.496 245.3311
3 ELS Master ELS Slave
4 Cam Equation Coeffecients Value
5 M (F1) 1
6 N (F2) 1
7 H (F3) 1
8 L (F4) 0
9 Slave or Master Phase Adjust
10 Sph (F7) 0
11 Mph (F8) 0
Module2 Module1 Sheet1 Sheet2

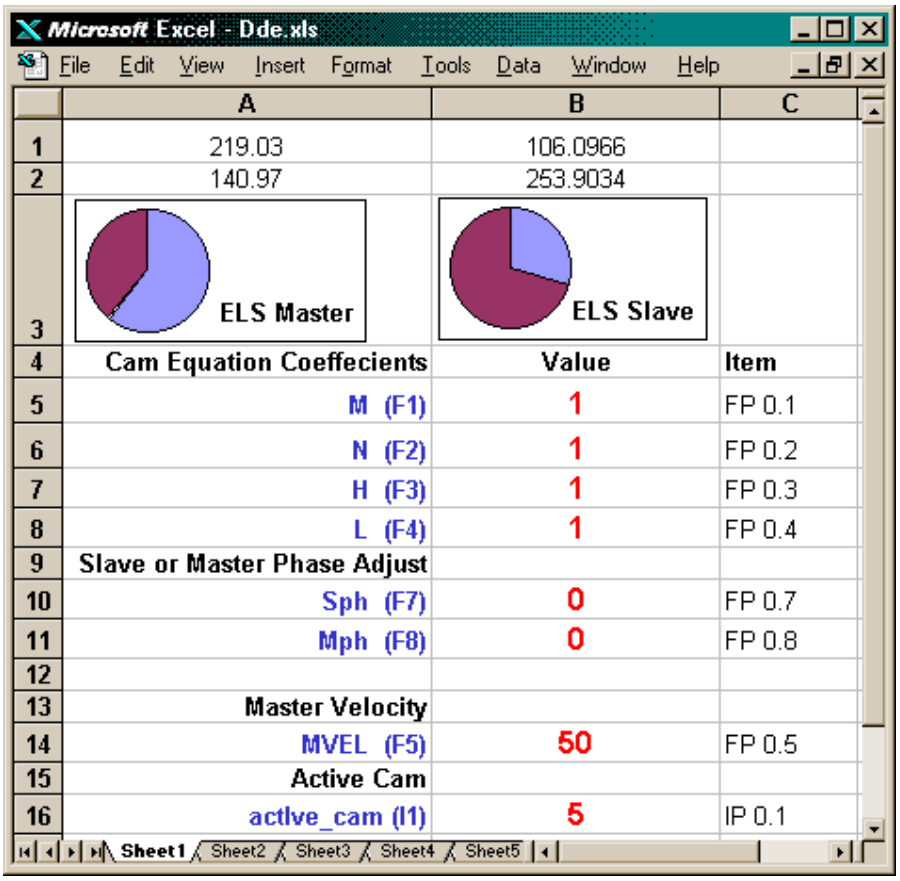
```
Function Request()                                          DDEInitiate ("Service Name", "Topic Name")
MYCLC = DDEInitiate("CLC_DDE", "SERIAL_0")

m = Application.DDERequest(MYCLC, "AP 1.032")               A-0-0032 Cam Slave Factor (M)
Worksheets(1).Cells(5, 2).Value = m
n = Application.DDERequest(MYCLC, "AP 1.031")               A-0-0031 Cam Master Factor (N)
Worksheets(1).Cells(6, 2).Value = n
h = Application.DDERequest(MYCLC, "AP 1.033")               A-0-0033 Cam Stretch Factor (H)
Worksheets(1).Cells(7, 2).Value = h
l = Application.DDERequest(MYCLC, "AP 1.035")               A-0-0035 Cam Master Position (L)
Worksheets(1).Cells(8, 2).Value = l
sph = Application.DDERequest(MYCLC, "AP 1.162")             A-0-0162 Cam Slave Phase Adjust
Worksheets(1).Cells(10, 2).Value = sph
mph = Application.DDERequest(MYCLC, "AP 1.151")             A-0-0151 ELS Phase Offset
Worksheets(1).Cells(11, 2).Value = mph

End Function
```

Figure B-4: Macro 1

Additional variables were added to column A to adjust the ELS Master velocity (F5) and the active CAM number (I1). Column C contains the **Item name** which corresponds with each program variable (F1-F8 and I1). The DDEPoke command in **Macro 2** references the worksheet for the DDE **Item name** and value for each variable.


Microsoft Excel - Dde.xls
File Edit View Insert Format Tools Data Window Help
ELS Master
ELS Slave
Sheet1 Sheet2 Sheet3 Sheet4 Sheet5

	A	B	C
1	219.03	106.0966	
2	140.97	253.9034	
3			
4	Cam Equation Coeffecients	Value	Item
5	M (F1)	1	FP 0.1
6	N (F2)	1	FP 0.2
7	H (F3)	1	FP 0.3
8	L (F4)	1	FP 0.4
9	Slave or Master Phase Adjust		
10	Sph (F7)	0	FP 0.7
11	Mph (F8)	0	FP 0.8
12			
13	Master Velocity		
14	MVEL (F5)	50	FP 0.5
15	Active Cam		
16	actIve_cam (I1)	5	IP 0.1

When **Macro 2** is executed the data in the value column will be written or "poked" to the variables defined by the corresponding **Item names**. This allows the user to see how different values will alter the performance of the slave axis with respect to the master. The DDEPoke command uses the following syntax:

Application.DDEPoke *MYCLC*, *Item Name*, *Value*

MYCLC is defined in the *DDEInitiate* command and includes the DDE **Service name** and **Topic name.**

```
Function Poke()

MYCLC = DDEInitiate("CLC_DDE", "SERIAL_0")

Application.DDEPoke MYCLC, Worksheets(1).Cells(5, 3).Value, Worksheets(1).Cells(5, 2)
Application.DDEPoke MYCLC, Worksheets(1).Cells(6, 3).Value, Worksheets(1).Cells(6, 2)
Application.DDEPoke MYCLC, Worksheets(1).Cells(7, 3).Value, Worksheets(1).Cells(7, 2)
Application.DDEPoke MYCLC, Worksheets(1).Cells(8, 3).Value, Worksheets(1).Cells(8, 2)
Application.DDEPoke MYCLC, Worksheets(1).Cells(10, 3).Value, Worksheets(1).Cells(10, 2)
Application.DDEPoke MYCLC, Worksheets(1).Cells(11, 3).Value, Worksheets(1).Cells(11, 2)

Application.DDEPoke MYCLC, Worksheets(1).Cells(14, 3).Value, Worksheets(1).Cells(14, 2)

Application.DDEPoke MYCLC, Worksheets(1).Cells(16, 3).Value, Worksheets(1).Cells(16, 2)

End Function
```

Figure B-5: Macro 2

B.2 Wonderware

In order for Wonderware to communicate with the CLC, a DDE link between the two must be created. The link, or DDE Access, tells Wonderware what Windows application to use (clc_dde.exe server) in order to communicate with the CLC. This application must be running in order for Wonderware to communicate with the CLC.

1. To establish a DDE link:
   A. Choose DDE Access Names under the Special Menu in Intouch Development. The DDE Access Name Definition window will open
   B. Press the Add button:

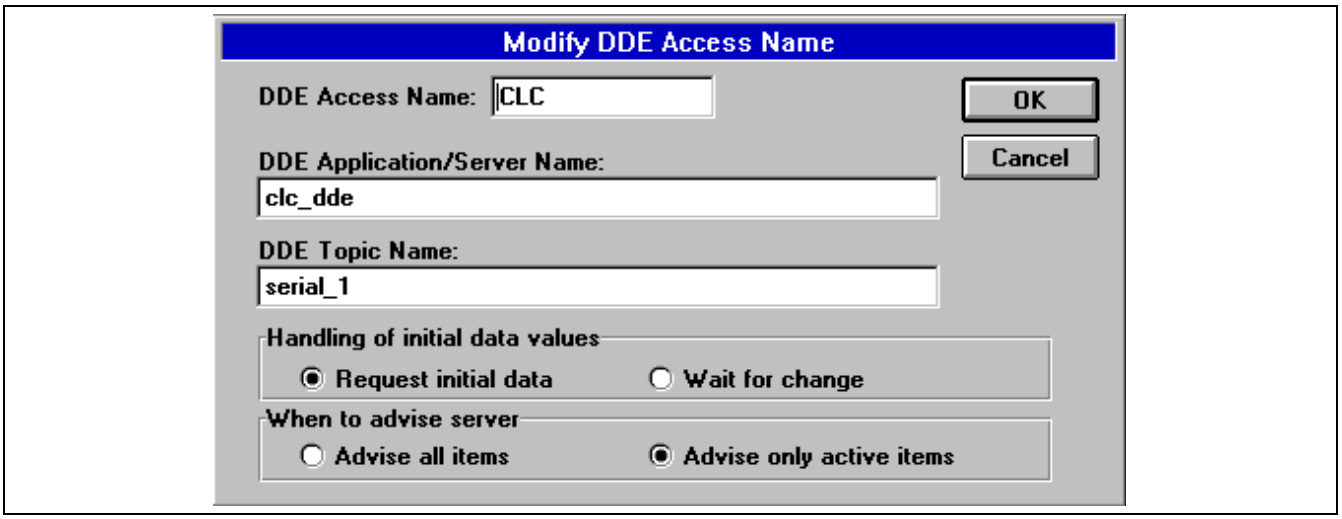


Figure B-6: DDE Access Name

- DDE Access Name:

  The DDE Access Name can be any name you choose.

- DDE Application/Server Name:

  For the CLC, this is the clc_dde server (clc_dde.exe) provided in the Visual Motion toolkit. It is not necessary to use the *.exe* extension. It is good practice, however, to include the path for the clc_dde server in the DOS Path statement and to configure Wonderware to launch the clc_dde server. If Wonderware is in your Windows Start-up group, then the clc_dde server application should also be in that group. Since Windows launches applications in the startup group from left to right, the clc_dde server icon should be to the left of the Intouch icon.

- Topic Name:

  The topic name will depend upon the method of communication between the computer and the CLC. The following items describe the different methods of communication.

  Serial Communications

  If you are communicating with the CLC (D, P, or V) via the computer's serial port, the topic name will be "serial_*x*" where *x* is the CLC device number, card parameter C-0002. The default for a CLC-D and CLC-P is device #0. The CLC-V is switch selectable via the Mode switch on the front of the card.

  PC Backplane Communications

  If you are communicating with the CLC-P via the PC backplane, the topic name will be "isa_*x*" where *x* is the PC address of the card.

  VME Backplane Communications

  If you are communicating with the CLC-V via the VME backplane, the topic name will be "xycom_*x*" where *x* is the VME card address set by the Mode switch on the front of the CLC-V.

  If you are still not sure of the topic name, it can be found in the CLC server application after Visual Motion has established communications with the CLC. To find the active topic name, open "Server Configuration" under the Settings menu in the clc_dde server application. The topic name is found in the "CLC Status Display" box.

Tagnames

To display a CLC parameter, variable, etc. in Wondware, the following type of tagname can be used:

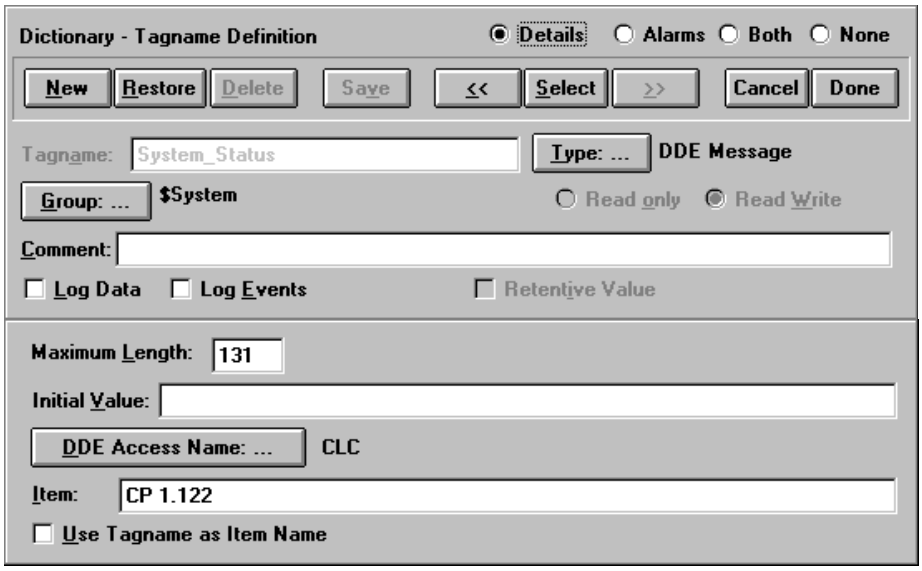


Figure B-7: Tagname for Displaying Parameters

The tagname in Figure BBB-777: Tagname for Displaying Parameters is labeled *System_Status*. It is type DDE Message, since it is only displaying the parameter. DDE Access name is “CLC”, from Figure BBB-666: DDE Access Name.

The item field requests the parameter or variable to display. In Figure BBB-777: Tagname for Displaying Parameters, the item requests CLC Card Parameter 122, system staus. Other examples:

Parameters	**Example**	
Card	CP 1.122	Card param. 122 System Status Message
Task	TP 1.123	Task A param. 123, Task Status Message
Drive	DP 2.95	Drive 2, param 95 Drive Status Message
Axis	AP 3.4	Axis 3, param. 4 Axis Options
Variables	**Example**	
Floating Points	FP 0.12	Active Program, float # 12
Integers	IP 1.5	Program #1, integer #5
Global Floats	GF 0.1	Active Program, global float #1
Global Integers	GI 2.2	Program #2, global integer #2

To write to a CLC variable, parameter, etc, the following type of tagname can be used:

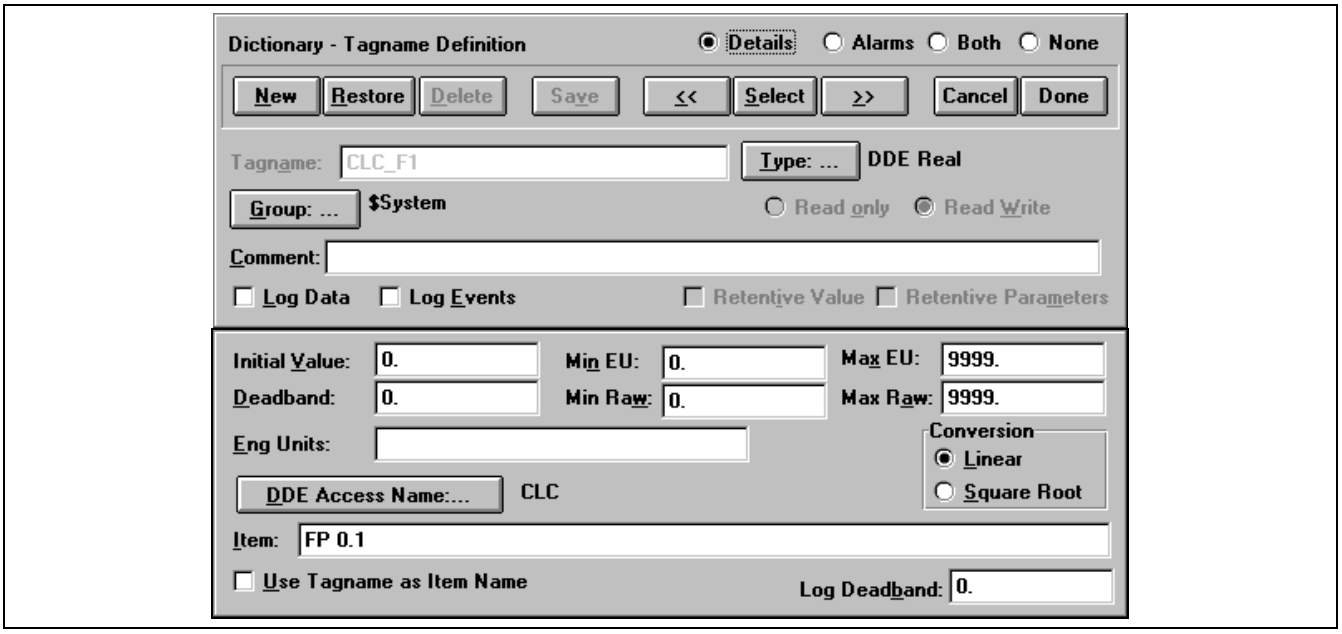


Figure B-8: Tagname for Writing to Variables

The tagname in Figure BBB-888: Tagname for Writing to Variables makes floating point variable #1 of the active CLC program a DDE real variable. The value of FP 0.1 can be changed in Wonderware by changing the value of the tagname, CLC_F1.

To change a bit in a CLC register:

Configure a tagname as a Type DDE Integer, min EU -99999, min Raw -99999

max EU 99999, max Raw 99999

Tagname Reg_100

Item RD 0.100

The above tagname will write to register 100. To change a specific bit in that register, configure a pushbutton as shown below.

Object type: Button

Button Type: User Input, Discrete

Tagname: Reg_100.00

The above button has two states, on and off. When in the on state, register 100 bit 1 will be set to 1. When in the off state, that bit will be set to 0. To write to bit 2, change the tagname to Reg_100.01, bit 3 = Reg_100.02, bit 4 = Reg_100.03, ... bit 16 = Reg_100.15.

C ELS Configuration

C.1 Electronic Line Shafting (ELS) Overview

Electronic Line Shafting or ELS is used to synchronize one or more slave axes to a master axis. An ELS master can be a real or virtual axis. A real master can be another axis in the system, or an external feedback device such as an encoder. A virtual master is a command generated by the CLC. Each slave axis can use either **velocity**, **phase** or **cam synchronization**. An ELS also includes the capability to jog each axis synchronously or independently, and to adjust phase offset and velocity while the program is running.

Since the master position is transmitted using SERCOS, all slave axes act upon the new master position at the same time thus insuring tight synchronization between all axes whether there are 2 or 25 slave axes.

The ELS Axis Configuration icon in VisualMotion is used to define the master and slaves in an ELS system but it doesn't configure all the parameters that are necessary for an ELS application. Some parameters must still be configured through the Parameters Overview list or through parameter transfers in the user program.

C.2 ELS Masters

The ELS master position and velocity, for all master types, can be monitored from System Parameters C-0-0157 and C-0-0158. System Register 30 is a status register for the ELS master. This register applies to all master types, but is used primarily for the virtual master. Status bits include: Master Type - Virtual/Real; Virtual Master enable; Virtual Master at programmed speed; Master Direction - Positive/Negative; and Master Stopped.

Real Masters

If another axis in the system is the real ELS master, then the master is defined as **Follow Axis Feedback**. The master axis must be defined separately using the single axis configuration icon as single axis, rotary mode. The single axis gear ratio and zero velocity window are used here to configure the master.

If a **master encoder** is to be used, then the master is defined as either a **Real Master through SERCOS** or **Daisy-Chained Real Master**. If only one drive contains a DFF master encoder card, then it is a real master through SERCOS. If all slave axes contain DFF cards, then it is a daisy chained real master.

NOTE: *GPS-5 and DIAX03 drives don't support Daisy-Chained Masters*.

System Parameters C-0-0159 (Real Master Zero Velocity Window), C-0-0161, C-0-0162 (Real Master Ratio Input & Output) and C-0-0163 (Real Master Filter Time Constant) should be configured through the Parameters Overview screen for these master types.

The zero velocity window determines when the master is stopped for diagnostic purposes or for a mode change. The input and output ratio are used as a "gear ratio" for the master encoder –e.g. it takes 4 revolutions of the master encoder to complete one "cycle" of the master. The displayed master position here is always rotary with a modulo of 360, regardless of the real master ratio. The filter time constant is used to filter out master encoder oscillations in the system caused by external disturbances.

Virtual Masters

The **Virtual Master** is an axis internal to the CLC (*Axis number 0*). It is controlled using the acceleration, velocity, stop and go icons in the user program. Its units are degrees, RPM, and rads/sec^2 for position, velocity and acceleration, respectively. To "home" the virtual master, transfer a value of "0" to C-0-157 using a parameter transfer in the user program. When an E-Stop is issued by the CLC (reg.1 bit 3 =0), the virtual master decelerates to a stop using parameter C-0-156 –Virtual Master E-Stop Deceleration.

Second Master

A second master can also be configured for ELS systems and the CLC can switch between these masters on the fly. The second master is configured as Master 2 and can be a virtual master, real master encoder, or follow axis feedback. Currently, there are no icons in VisualMotion to configure Master 2 so it must be done manually through the parameters "Overview" or through parameter init statements in the user program.

C.3 ELS Slaves

After the master has been configured, the slave(s) must be setup. The following master/slave synchronization options are available:

Phase Synchronization

Velocity Synchronization

CAM Synchronization

Each one of these ELS modes can also be used with Dynamic synchronization

When a slave is not in synchronization with the master, it is in secondary mode. Secondary mode can be either single axis -- issue position commands to the axis -- or velocity – axis moves at a constant velocity. *See Slave Secondary Mode* (Non-Dynamic Synchronization).

Phase Synchronization

Phase synchronization maintains the same relative position among axes, but adjusts the lead or lag of the slaves to the master in terms of degrees. It is used when the positions of axes are most critical. A slave configured as **ELS Phase** follows the position of the master linearly at the ratio specified in the *Turns* setup within the ***ELS Axis Configuration icon*** in VisualMotion.

ELS samples an absolute (one turn or multi-turn) master position, transmits that position to the slave axes and the slave axes execute the equation:

$$\phi_{slave} = \phi_{master} * (K_{slave} / K_{master}) + phase\_offset$$

Slave position = sampled master position * (slave turns/ master turns) + phase offset

If the turns ratio (Ks/Km) is ¼, then the master must make 4 revolutions for the slave to complete 1 revolution. The position units for a phase slave are degrees and the slave modulo is always 360 * Ks/Km. To insure that there is no round off error in the turns ratio, the ratio must be entered using integer values.

One of the features of a phase slave is that it can be at a position offset from the master. This “phase offset” is by default automatically initialized when the slave is commanded to synch to the master – e.g. if the turns ratio is 1/1, the master is at position 100 and the slave is at position 120, the phase offset is initialized to 20.

The phase of the slave can be advanced (or retarded) while the system is in motion. The ***ELS adjust icon*** is used to move the slave to a commanded offset. When the phase adjust command is issued the axis performs a move on top of the current motion to the commanded offset position. An “Absolute” offset will move the axis so that the phase difference between the master and slave is equal to this value – e.g. if the turns ratio is 1/1, the master is at 100 and the slave is at 70, and an absolute phase offset of 20 is commanded, then the slave moves 50 degrees to 120.

An “Incremental” offset will add the specified number of degrees to the current phase difference – e.g. if the turns ratio is 1/1, the master is at 100 and the slave is at 70, and an incremental phase offset of 20 is commanded, then the slave moves 20 degrees to 90.

Phase Synchronization Features

With Diax03 drives, the **phase adjust move** can be controlled by either the CLC or by the drive. Axis parameter A-0-0164 (ELS Options) bit 1 determines if the phase offset will be controlled by the CLC (=0) or by the drive (=1).

If the phase offset is controlled by the drive it can use a **drive filter** or **dynamic synchronization**. If a drive filter is used, then the user specifies only the time in which to make the phase adjust. The drive uses parameter P-0-0060 (ELS Phase Adjust Time Constant) as the time constant for a low pass filter to calculate a jerk limited move profile for the phase offset.

Absolute Synchronization

The Diax03 drives can also be configured so that the phase offset is not automatically initialized at the time of synchronization. By setting bit 6 of Axis Parameter A-0-0164 to 1, the CLC initializes the phase adjust to the value in Axis Parameter A-0-0151 (Programmed Phase Offset). If Axis Parameter A-0-0151 is set to 40 in the above example, the slave would perform a 20 degree phase offset to position 140 when commanded to sync to the master.

Axis Parameter A-0-0164 can be set only in Parameter Mode (Phase 2). To set this parameter in the user program, use the "ParamInit" icon. Axis Parameter A-0-0151 can be set at any time with either a Parameter Transfer icon or with the ELS Adjust icon.

	CLC Controlled Phase Offset	**Drive Controlled With Filter**	**Dynamic Synchronization**
Axis Parameter A-0-0164- bit 1	0	1	1
ELS Phase Adjust Time Constant	A-0-0155	P-0-0060	N/A
ELS Phase Adjust Velocity	A-0-0153 *used only when A-0-0155* = 0	N/A	P-0-0143
ELS Phase Difference	P-0-0034	P-0-0034	P-0-0034
Non-Shortest Path Phase Adjust	A-0-0004 bit 10 = 1	N/A	P-0-0154

Relative Synchronization

If A-0-0164 bit 6 = 0, the CLC sets the phase offset to the difference between the master and slave position and no move is made.

Velocity Synchronization

Velocity synchronization relates slave axes to a master in terms of rotational rate. It is used when axis velocities are most critical, as in paper processing operations in which two or more motors act on a single piece of fragile material.

A slave configured as **ELS Velocity** follows the velocity of the master at the ratio specified in the Turns setup. In this mode, the master velocity is calculated from successive master position samples and the slave is commanded to that velocity using the equation:

$$V_{slave} = V_{master} * ((1 + fine\_adjust) * (K_{slave} / K_{master}))$$

The "fine adjust" value in this case is an additive velocity expressed in percent of current speed. This can be used to fine adjust the following ratio. The limit on the fine adjust ratio is $\pm$ 300 %.

See Parameters P-0-0083, A-0-0150 and A-0-0159 to change the fine adjust value.

CAM Synchronization

Cam synchronization is used when custom position, velocity or acceleration profiles are needed at a slave axis. These special profiles are developed at the slave by sending position commands every SERCOS cycle.

A cam is an (x, y) table of positions that relate a master axis to a slave. Cams can be stored on the CLC or on the digital drive. CLC cams have more adjustment options and can work with any SERCOS drive. Drive cams are more efficient and can be applied to more axes. The same programming commands and utilities are used for both drive-resident and CLC-resident cams.

The velocity profiles are generated from a cam (x,y) table that contains angular position data for the master axis and the corresponding cam position. A *normalized* cam commands 360 degrees of rotation for every 360 degrees of the master. It contains 1024 points where 0.0 <= x <= 360° and 0.0 <= y <= 360°.

It is possible that the position of the master for a given SERCOS cycle does not match exactly with one of the x points from the table. In this case a linear interpolation between the two table points estimates the unknown position.

Dynamic synchronization

Dynamic synchronization is available in DSM2.3-ELS-04VRS and GPS-05VRS firmware. In ELS phase and drive cam modes, each slave axis can independently ramp up to synchronization with a running master and lock on to its position. The drive controls the ramp and the phase offset profile.

For all ELS modes:

The ramp starts when the CLC commands the drive to switch into synchronization mode. The slave drive then accelerates at a parameter-determined rate to match the synchronous speed.

For phase synchronization and electronic cam modes:

At the end of the ramp, a phase offset will be performed automatically before the axis is synchronized. This compensates for any inaccuracy caused by the ramp being started too late or the velocity changing during the ramp.

While the slave axis is synchronized, phase offset can be adjusted at any time with a profile that has adjustable velocity and acceleration (like a single-axis move).

See the following Parameters for more information:

C-0-1012 Master Synchronization Acceleration
C-0-1013 Master Synchronization Time Constant
C-0-1014 Master Synchronization Time Constant

A-0-0151 Programmed Phase Offset
A-0-0164 ELS Options

S-0-0228 Position Synchronization Window

P-0-0034 Additive Position Command Actual Value
P-0-0142 Synchronization Acceleration
P-0-0143 Synchronization Velocity
P-0-0154 Synchronization Direction
P-0-0155 Synchronization Mode

Operation of Dynamic Synchronization

Ramp-up to synchronous mode

Set parameter P-0-0142 and P-0-0143. Since these parameters are stored through the service channel, this should be done only at the start of the program or when the values need to be changed.

Set the phase offset position with the ELS adjust user program command. At the end of the ramp, a phase offset move to this position will be performed.

To start ramping to the synchronous velocity, execute the ELS mode change user program command with the 'Sync to Master' option. The drive will start the ramp-up/lock-on procedure, based on the phase offset and the synchronization acceleration.

Ramp-down to single-axis mode

Execute the ELS mode change command with the 'Single Axis' option. To immediately stop the axis, execute the Axis Stop command. To stop to a position, execute the Axis Move command with the absolute position desired.

NOTE: *If the axis stop command does not follow the axis mode change command, the axis will move to the last stored target position. Therefore, it is necessary to add a stop command or a new move command after switching into single-axis mode.*

Synchronization Status

Axis status register bit 4 'Phase Adjusted' indicates the phase offset status. If the difference between the drive's current position and its synchronous position (output of ELS or cam equation) is within the synchronization window (S-0-0228), the bit is set to (1). Otherwise, it is (0).

ELS Master Synchronization

The ELS Master Synchronization feature allows synchronization between a virtual master and a real master on-the-fly. This can be used for applications in which the slaves need to be stopped or started independent of the real master, while still following cam or ELS profiles. An example of this is an in-die press transfer application.

Implementation

Dynamic master synchronization is activated by either of two methods:

- A transition from 1 to 0 of the Virtual/nReal control bit (R29 bit 5)
- The ELS_mode command with axis 0, mode 'sync to real master', and real master number (1 or 2) selected. *Note: In first GPS-6 test releases, this is not implemented*.

After synchronization is activated, the virtual master will accelerate at a parameter-defined rate to reach synchronous speed with the real master. The starting position of the virtual master acceleration is automatically calculated by the CLC so that (ideally) the virtual and real master positions will be equal when the synchronous velocity is reached.

At the end of the acceleration, when the virtual and real master velocities are matched, a comparison is made between the virtual and real master positions. A phase offset equal to the position difference is then performed automatically before the master positions are synchronized. This compensates for any inaccuracy caused by the ramp starting point, or a change in the real master velocity during the ramp. The phase offset is performed as a move controlled by a first-order filter superimposed on the velocity of the virtual master. The phase offset and error elimination process is transparent at the Visual Motion/application programming level.

See the following parameters for more information:

C-0-1012	Master Synchronization Acceleration
C-0-1013	Master Synchronization Time Constant
C-0-1014	Master Synchronization Velocity Window

Master Synchronization Procedure Implementation

Given:

A	Slope of the velocity ramp (C-0-1012)	(deg/sec/sec)
Pr	Current position of real master	(0 - 360 deg)
Pv	Current position of virtual master	(0 - 360 deg)
Sr	Speed of real master	(deg/sec)
Sv	Speed of virtual master	(deg/sec)

Compute:

Rt	Ramp time	(sec)
Fr	Calc. real master position at end of ramp	(0 - 360 deg)
Fv	Virtual master at end of ramp	(0 - 360 deg)
Pf	Difference between final master positions	(0 - 360 deg)
Pw	Threshold for starting the ramp	(0 - 360 deg)
Pe	Phase error	(0 - 360 deg)

Rd Ramp distance (deg , no modulo)

Step 1: Find the ramp starting position

Calculate the time it takes to ramp up to speed, and the distance traveled during the ramp:

Rt = (Sr - Sv) / A

Rd = 0.5 * A * Rt*Rt

Keep calculating the end positions of the real and virtual masters, assuming constant real master velocity during the ramp:

Fr = Pr + (Sr * Rt)

Fv = Pv + Rd

Pf = Fr - Fv

The threshold **Pw** is calculated automatically based on the distance moved in two position updates, dependent on the real master velocity. Ideally, the virtual master will be slightly behind the real master at the end of the ramp, to prevent a move in the reverse direction.

When **Pf < Pw**, start the acceleration ramp.

Step 2: Velocity Synchronization

Continue ramping the virtual master until the velocities of the real master and the virtual master intersect. When the sign of the differences between the real masters changes, velocity synchronization is complete.

During velocity synchronization, always monitor the real master velocity to be sure it is within the velocity synchronization window (C-0-1014). If the real master velocity changes in either direction by more than this window, the error “510 ELS Master Synchronization Error” is issued. The virtual master and the drives will follow the error reactions selected in parameters C-0-0009 and A-0-0004.

At the end of the ramp, the phase error is:

Pe = Pr - Pv

Step 3: Position Synchronization

Start following the velocity of the real master. Superimpose a move of distance **Pe** onto the master position, using a first-order filter. This will normally be a very small move, at a low additive velocity. It is assumed that the real master's speed will be constant during the phase offset move.

When the phase offset is complete (after the time specified in parameter C-0-1013), set register 30, bit 5 (Virtual/nReal) to (0), indicating that synchronization is complete. Start following the position of the real master.

Slave Secondary Mode (Non-Dynamic Synchronization)

When a program begins execution, either at the beginning of a cycle or after an error, all slaves are in their secondary mode. They will not start following the master until a synchronize command is issued through the “ELS Mode Change” icon. When this command is issued, the slaves establish a relative synchronization to the master. If the master is moving when the synch command is issued, the slaves do not ramp up to the master speed (this feature, called Dynamic Synchronization, is available with version 4 of the Diax03 drive firmware).

The master should either be at standstill when the command is issued or the drives should be in velocity mode with a velocity equal to that of the master. When switching from synch mode to secondary mode, the axis does not ramp to a stop either. If switching to single axis positioning secondary mode, then the velocity is immediately commanded to 0. If switching from synch to velocity secondary mode, then the drive axis will continue moving at constant velocity equal to the last sampled master velocity. To stop the axis, either the “Stop” icon must be issued or a velocity of 0 commanded through the “Velocity” icon.

Summary of ELS Parameters and Registers

Master

Setup/Control	**Location**
Zero Velocity Window	C-0-0159
Real Master Ratio Input	C-0-0160
Real Master Ratio Output	C-0-0161
Real Master Filter Time Constant	C-0-0163

Status	**Location**
Position of master (fmaster) - degrees	C-0-0157
Current Velocity (Vmaster) - rpm	C-0-0158
Virtual Master at Programmed Speed	Reg. 30 bit 7
Master Direction: Positive/Negative	Reg. 30 bit 9
ELS Master Stopped	Reg. 30 bit 10

Slave

Setup/Control	**Location**
Phase Offset Controlled by CLC/Drive	A-x-0164, bit 1
Phase Adjust Parameters	see previous table

Status	**Location**
Position of slave (f_{slave}) - degrees	S-0-0047
Velocity of slave (Vslave)- rpm	S-0-0036
Phase Adjusted	Reg. 31 - 38 bit 4
ELS Enabled	Reg. 31 - 38 bit 5
Secondary Mode Enabled	Reg. 31 - 38 bit 6
Phase Difference	P-x-0034

C.4 Cam Applications

A cam is an (x, y) table of positions that relate a master axis to a slave. ELS (Electronic Line Shaft) systems make use of cams when special position, velocity or acceleration profiles are needed at a slave axis. These special profiles are developed at the slave by sending **position commands** every SERCOS cycle.

Overview

Cams can be stored on the CLC or on a digital drive (Indramat DDS 2.2 or a similar DIAX03 drive). CLC cams have more adjustment options and can work with any SERCOS drive. Drive cams are more efficient and can be applied to more axes. The same programming commands and utilities are used for both drive-resident and CLC-resident cams.

A *normalized* cam commands 360 degrees of rotation for every 360 degrees of the master. It contains 1025 points, 1024 intervals, where

$0.0 \leq x \leq 360°$ and $0.0 \leq y \leq 360°$.

It is possible that the position of the master for a given SERCOS cycle does not match exactly with one of the x points from the table. In this case a linear interpolation between the two table points estimates the unknown position.

A master axis may turn in a positive or negative direction without loss of cam function.

Cam Commands

The following commands are used to initialize and adjust cams. ***Refer to Chapter 6. Icon Programming for more information*.**

Cam

Cam Adjust

Cam Build

ELS

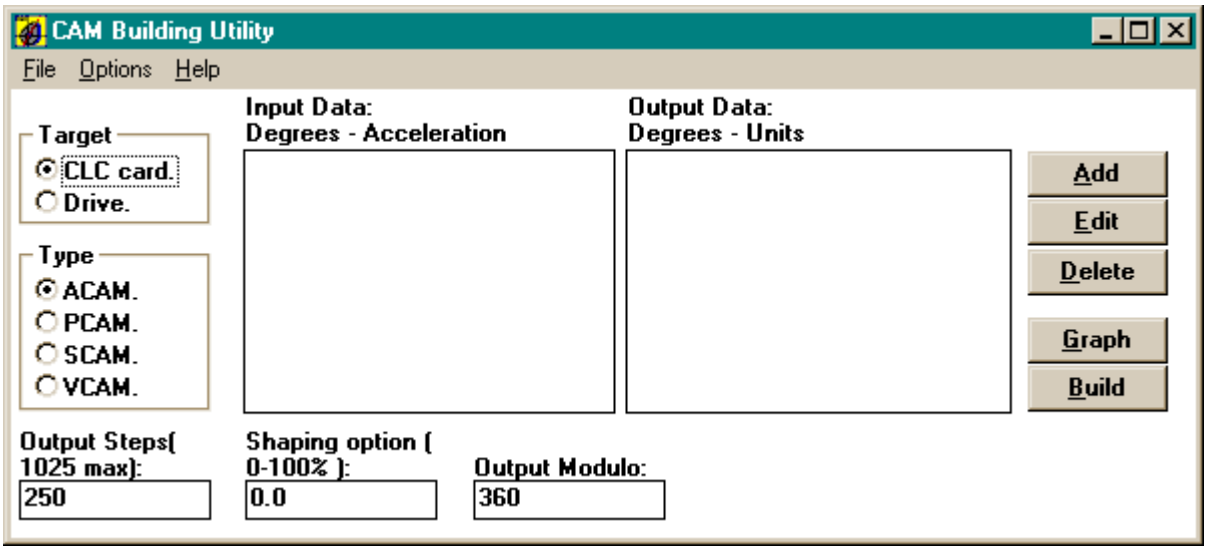
ELS Mode

C.5 Building Cam Tables

There are two ways to build a cam within VisualMotion:

The Cam Build Icon (***See Chapter 6. Icon Programming***).

The CAM Building Utility (***See Chapter 4. VisualMotion Menu Commands***). Refer to the Cam Builder Selection under the Tools Menu.



A spread sheet can also be used build a Cam Table. In the spreadsheet method, the equations that define the cam can be developed and executed to generate the xy table of positions. Although the xy table need not be normalized, it is a good idea to use the normalized form. The cam shaping parameters are easier to use when the cam table is normalized. ***See Section Cam Equation***.

Cams enter the CLC as text files in CSV format, the kind most spreadsheets generate.

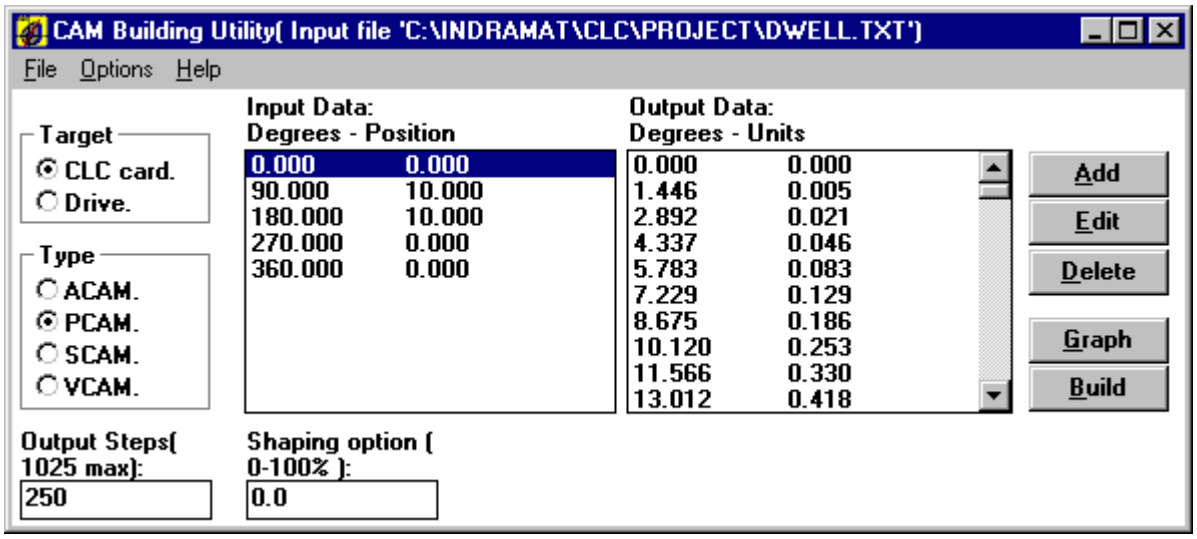
If the CSV file contains less than 1025 points, an algorithm within the CLC fills in the missing points. As a rule, a CSV cam file should contain at least 200 points, anything less than that does not sufficiently define the cam and unexpected results may occur.

PCAM Cam Build Type

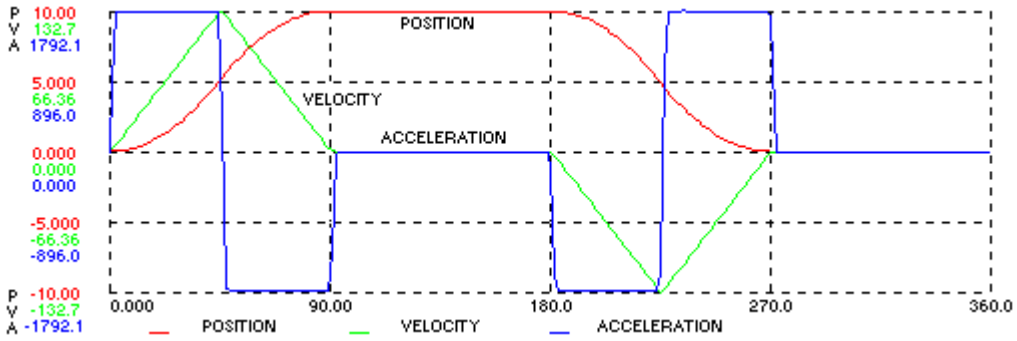
The PCAM build type accepts input in the form of a table of target **positions**. The utility generates smooth table values between the targets, each target being viewed as a separate accelerate/decelerate type motion. Thus, a start/stop type motion is embedded in the output cam table for each target specified in the input file.

It is a good idea to limit the number of targets to less than 10 and to keep sufficient distance between targets to allow for a fully formed motion profile between them. A rule of thumb is to have at least 20 degrees between targets.

Example:



With this position table the PCAM build type will generate the following cam table:

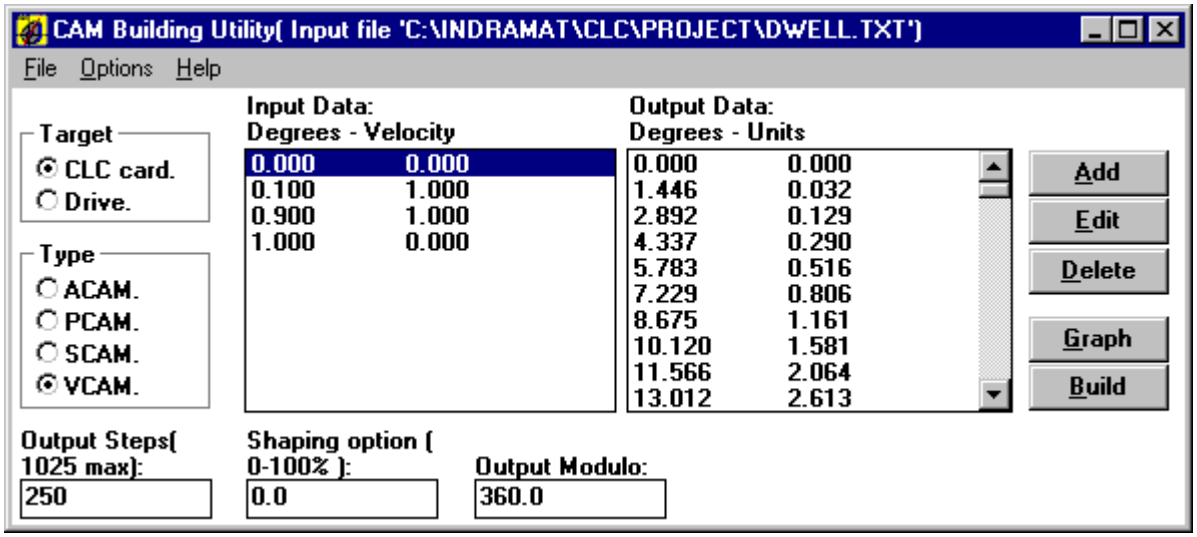


Since all transitions in position (such as above - 0 to 10 degrees) use a start/stop type profile, the number of transitions must be limited and must not be too close together. Otherwise, the CAM motion will become jerky.

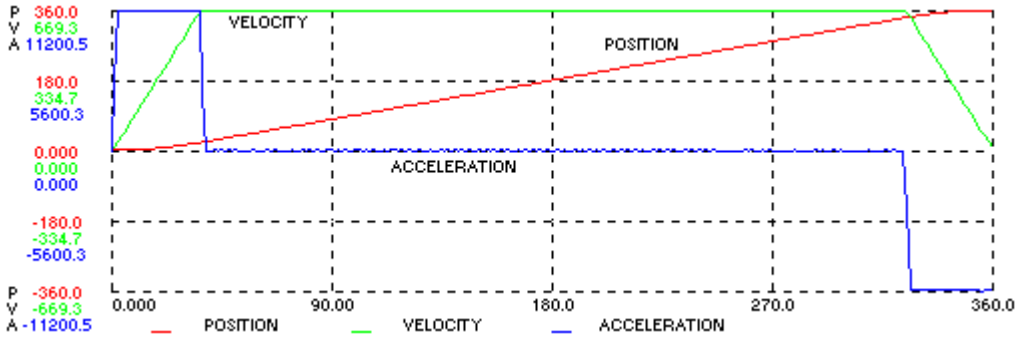
VCAM Cam Build Type

The VCAM build type accepts a **velocity** profile as input and outputs a normalized profile. The following examples uses **Time - Velocity** instead of **Degrees - Velocity** for the input data. Both units can be used to build output data.

Example 1:



With the velocity profile defined as above the VCAM build type will generate the following velocity profile curve:

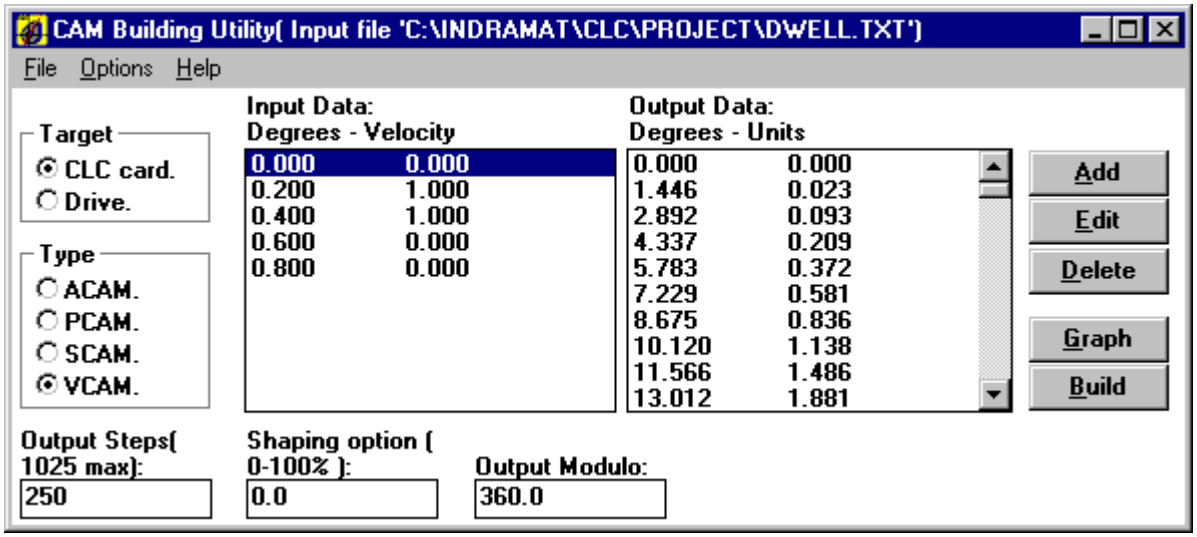


The total distance traveled by the slave is 360 degrees since the profile is normalized.

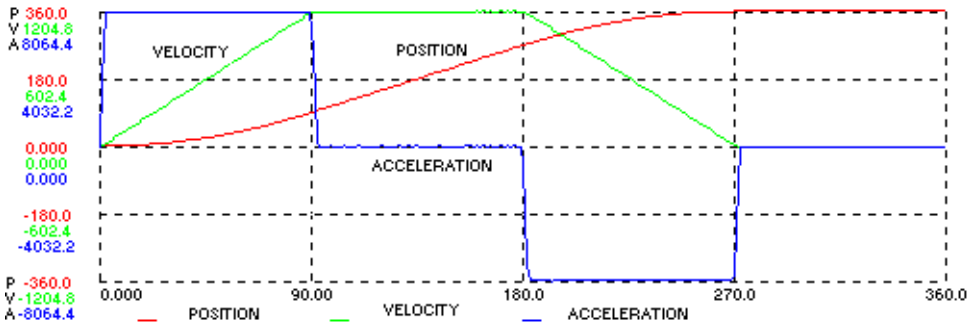
In CAMs that cycle back to a zero *position*, it is important that the total area under the *velocity* curve defined by the input file be zero, otherwise the CAM will gain or lose a position offset every cycle (*as illustrated in the PCAM example*).

Example 2 - Velocity Profile with Dwell:

In this next example the velocity profile of the cam contains a 25% of cycle dwell.



The resulting cam velocity profile follows:



By adding more lines of input data the *velocity* profile can be shaped to almost any degree.

The *velocity* profile of the cam will match the original input file when the master cycle time is 0.80 second.

As the master cycle time increases the cam profile will stretch over time but the total *position* commanded to the slave remains 360 degrees every cycle.

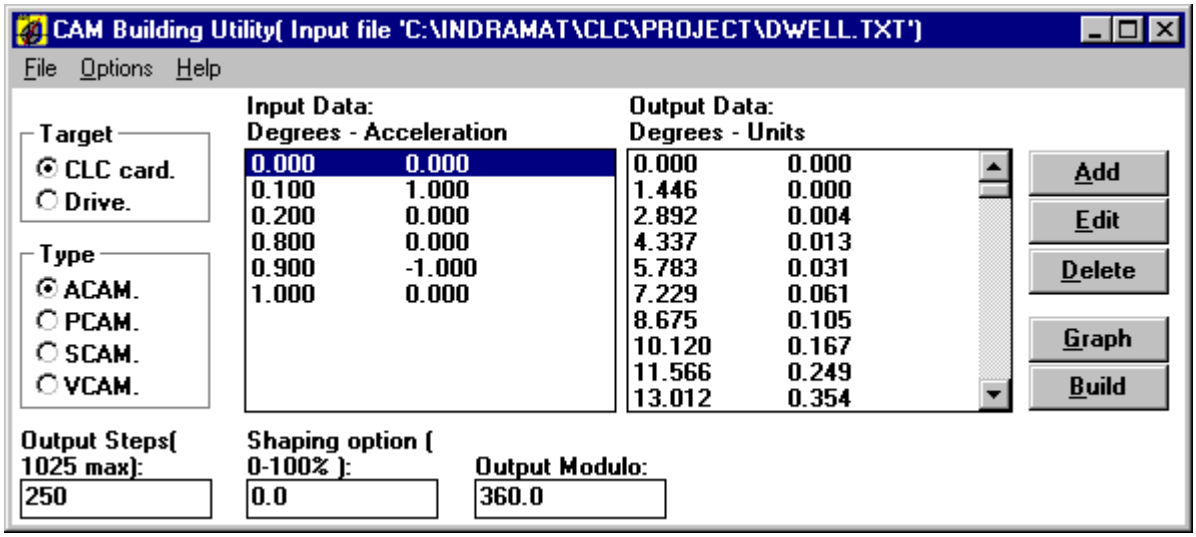
Likewise, as the master cycle time decreases the cam profile will shrink but the total *position* commanded to the slave remains 360 degrees every cycle.

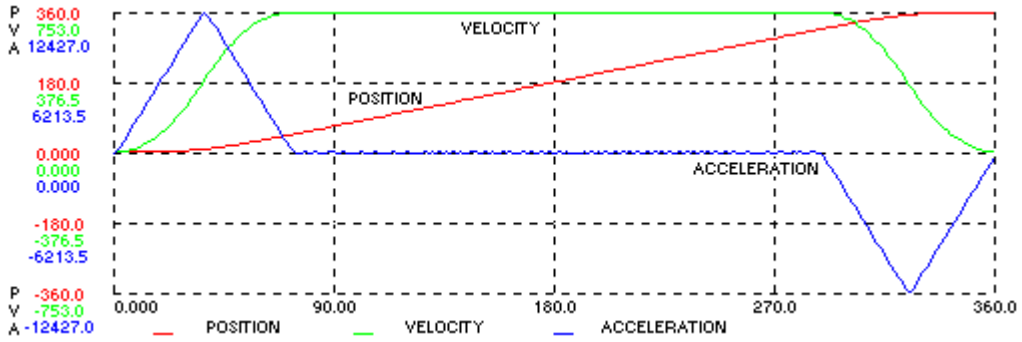
ACAM Cam Build Type

ACAM build type accepts an **acceleration** profile as input and outputs a normalized profile.

The following examples uses **Time - Acceleration** instead of **Degrees - Acceleration** for the input data. Both units can be used to build output data.

Example:





The distance traveled by the slave is 360 degrees since the cam is normalized.

It is important that the total area under the acceleration curve defined by the input file be zero, otherwise the cam will gain or lose speed every cycle and thus be unstable. The acceleration profile of the cam will match the original input file when the master cycle time is 1 second.

As the master cycle time increases the cam profile will stretch over time but the total position commanded to the slave remains 360 degrees every cycle. Likewise, as the master cycle time decreases the cam profile will shrink but the total position commanded to the slave remains 360 degrees every cycle.

SCAM Cam Build Type

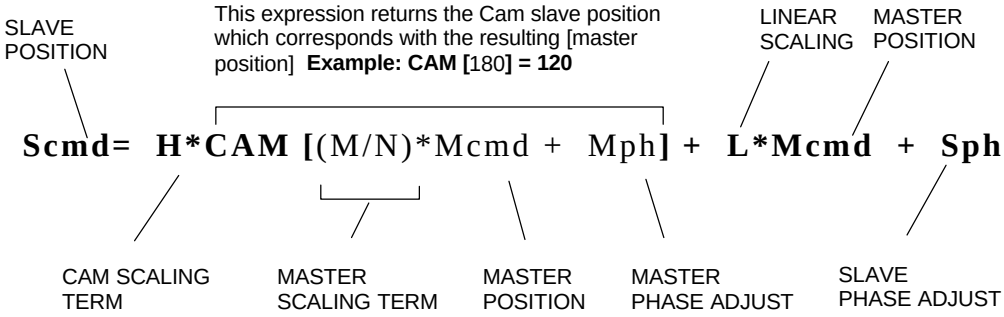
The SCAM build type accepts input in the form of target positions and outputs a cam by fitting these position to third order **spline** polynomials.

Use of this type of cam builder is more complex than the others due to the interplay between the target positions used.

It is possible to control cam shape and acceleration properties to almost any degree with this utility, but skill is required in choosing the proper target positions.

C.6 Cam Equation

Cams execute every SERCOS cycle and are governed by the following equation:


SLAVE POSITION
This expression returns the Cam slave position which corresponds with the resulting [master position] Example: CAM [180] = 120
LINEAR SCALING
MASTER POSITION
Scmd= H*CAM [(M/N)*Mcmd + Mph] + L*Mcmd + Sph
CAM SCALING TERM
MASTER SCALING TERM
MASTER POSITION
MASTER PHASE ADJUST
SLAVE PHASE ADJUST

Where,

Mcmd	Is the position command of the master (degrees)	C-0-0156
M	Multiplying scaling term of the master position	A-0-0032
N	Divisional scaling term of the master position	A-0-0031
Mph	Master Cam phase adjust (degrees)	A-0-0151
Sph	Slave Cam phase adjust (degrees)	C-0-3101….3108
CAM [ ]	Normalized 1025 cam table	A-0-0033
H	Cam scaling term	A-0-0035
L	Linear scaling term of master position	A-0-0162
Scmd	Commanded position to slave	A-0-0163

The effect a parameter has on motion can be seen by setting all other parameters to zero or one.

Consider H=0 and L≠0, Sph=0

Scmd = L*Mcmd.

L has the effect of introducing the position command of the master (**Mcmd)** into the cam equation, but in a linear way.

Setting **L** = 1/2 means that the slave axis makes one revolution to every two revolutions of the master, or the slave has 1/2 the speed of the master.

With **L** = 1 the slave and master move the same distance at the same speed. Thus, **L** effects the speed of the slave in a linear way.

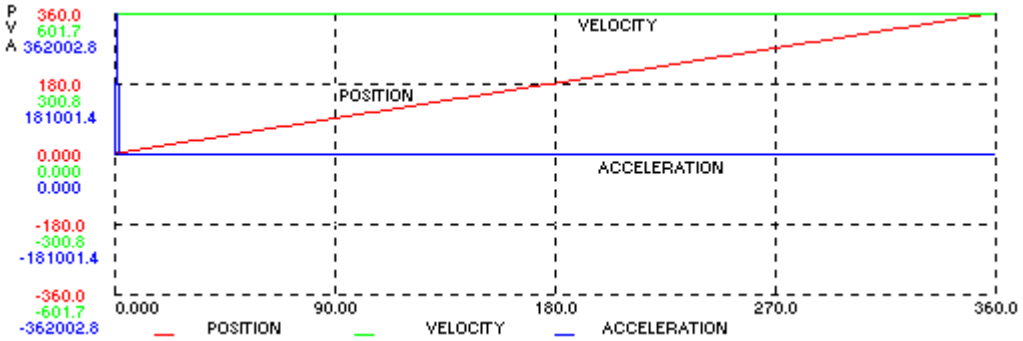
Consider H≠0, L=0, M=N =1 and Mph=Sph=0

Scmd = H*CAM[Mcmd].

What happens if H = 1? The slave axis is completely governed by the cam table. For each cycle of the master, the slave makes one cycle, or for each revolution of the master axis, the slave axis makes one revolution. The difference here is that the relationship between the slave and master is not linear.

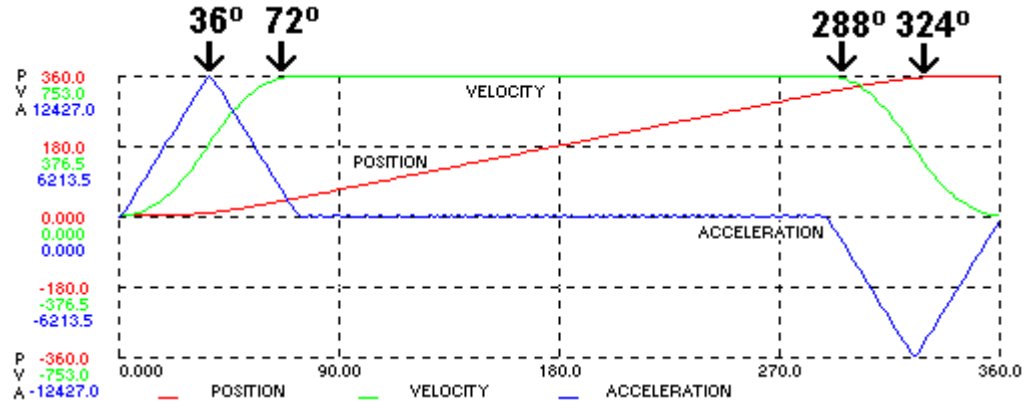
Consider the acceleration profile defined by the ACAM utility example See Section ACAM Cam Build Type. If the master axis is running at a constant velocity, what kind of motion is observed at the slave?

Master Axis running at a constant velocity between 0º and 360º



At the start of the cycle (master at zero degrees) the slave is at rest (stopped). When the master reaches 36 degrees the slave passes through peak acceleration and is speeding up rapidly. At 72 degrees the slave is at constant speed. The slave remains at constant speed until the master is at 288 degrees, here the slave starts to decelerate. Peak deceleration is achieved at 324 degrees, and by end of cycle the slave has stopped again.

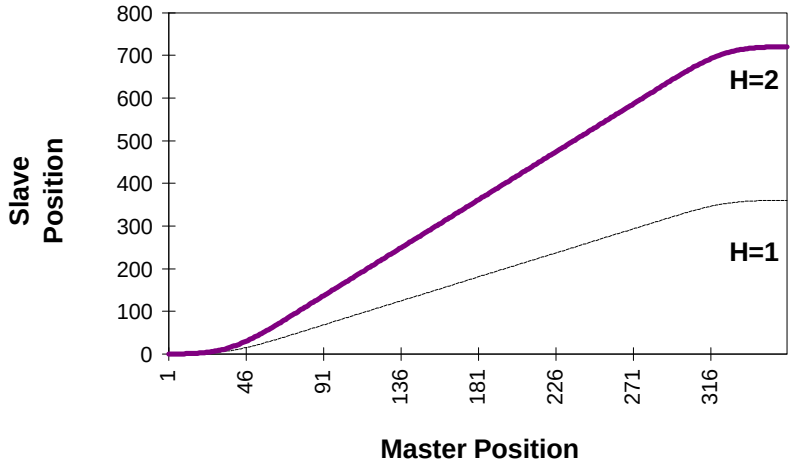
Slave Axis following the master above according to the ACAM profile



What distance has the slave traveled? Since the cam table is normalized we know that the **y** side (output side) of the table varies from 0 to 360 degree so the slave has traveled one revolution.

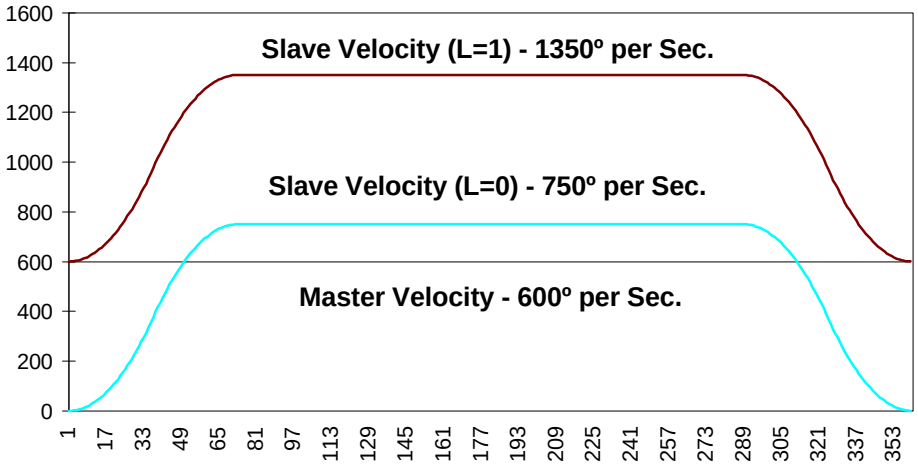
To summarize, the master axis rotates at constant velocity while the slave rotates at the same rate but does so by accelerating, traveling at constant speed, then decelerating to a stop. Thus the slave constantly starts and stops for every cycle of the master. This is perfect motion for indexing something into position once every cycle.

What happens if H = 2? The only thing that changes is the distance traveled by the slave. Instead of the slave making one revolution each cycle, as the master does, it makes two revolutions. So, H = 2 doubles the distance traveled by the slave. On the other hand, H=1/2 cuts the slave distance in half.



Consider H=L=1, M=N=1, and Mph=Sph=0?

Now the slave makes two revolutions each time the master makes one, but instead of starting and stopping, the slave rotates continually. In fact, the slowest the slave axis moves is at whatever the rate the master is.

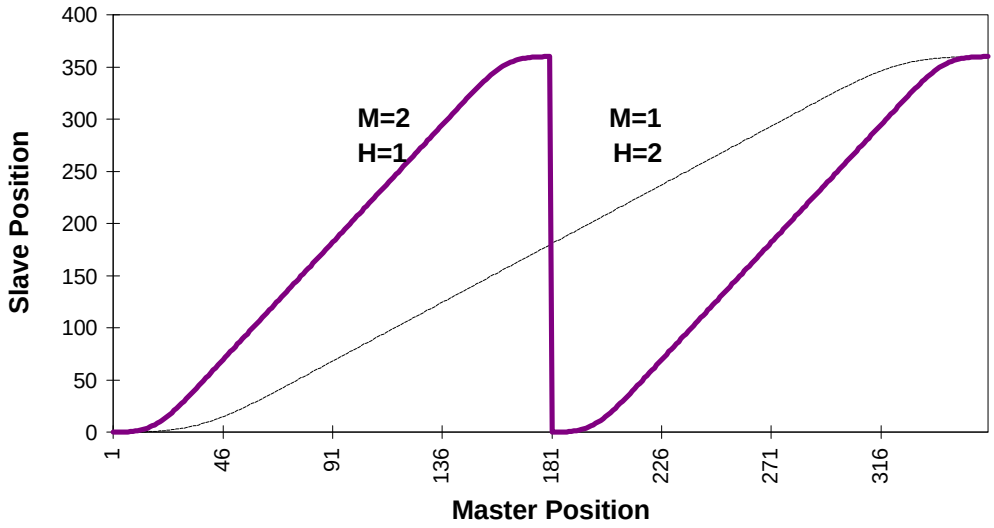


In other words, at the start and end of every cycle, the slave is rotating at the same speed as the master, but then speeds up and slows down to gain

another whole revolution every cycle. What happens is the velocity profile of the cam has been added to the line speed of the master.

Consider L=0, H=1, M=2, N=1 and Mph=Sp=0

As the master makes one revolution (360 degrees) the input to the cam table is doubled since (M/N)*Mcmd = 720 degrees, and so the cam table is traversed twice for every cycle of the master. The effect is that the slave advances two revolutions to each one of the master.



The effect M=2 and H=1 has is not the same as that of M=1 and H =2. The distance traveled by the slave is the same in both cases but the profiles used to travel that distance are vastly different.

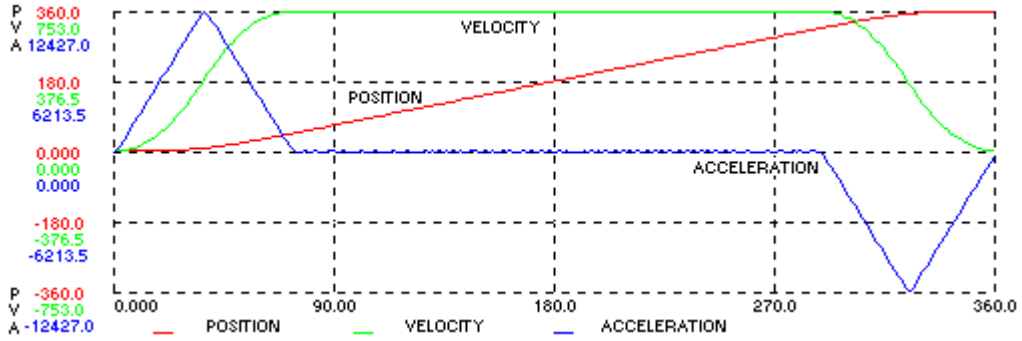
With M=2 and H=1, the slave starts and stops twice for each master cycle, however, for M=1 and H=2 the slave starts and stops only once. M=2 has the effect of putting out the cam profile twice per each master cycle, whereas H=2 has the effect of doubling the distance traveled during one cam cycle.

In essence, the effect M and N has is to define the number of cam cycles per master cycle.

Consider L=0, H=M=N=1 and Mph=90, Sph=90 degrees.

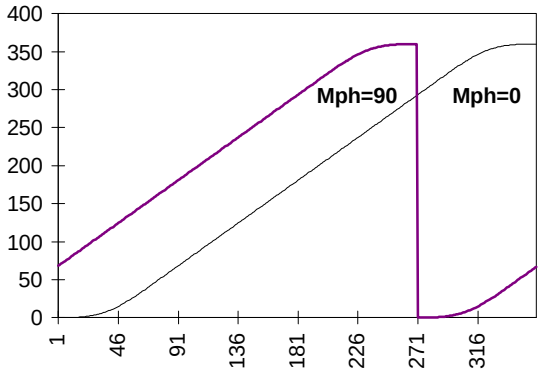
Finally, the phase adjust term needs to be considered. The effect here can be seen by stepping the slave through one cycle.

Slave Axis - ACAM profile (No phase adjust)



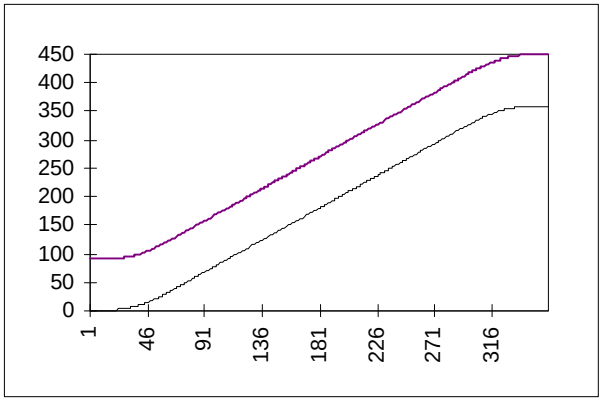
At the top of the cycle (master at 0º) the slave is 90 degree into the cam table. At that point, the slave is at constant speed and 18 degrees out of the acceleration stage. In other words, the cam profile is leading the master cycle by 90 degrees.

Slave Axis - Mph=90, Mph=0



The effect Mph has is to slide the cam profile right (-phase) or left (+phase) relative to a master cycle.

Slave Axis - Mph=90, Mph=0



The effect Sph has is to slide the cam profile right (-phase) or left (+phase) relative to a slave.

C.7 CLC Cam Alignment

It is possible to align a slave axis to the output position of the cam by using the align status, the current slave position, and single-axis positioning. Drives that do not support single-axis mode (i.e. the RAC) cannot use the alignment function.

The following procedure can be used in the CLC user program before switching an axis into synchronization mode:

1. Check the Aligned Status bit (axis status bit 8) to determine if a move needs to be performed.
2. Switch the axis to single axis mode using the ELS/MODE instruction.
3. Set the programmed velocity to the axis phase adjust velocity (A-0-0153).
4. Set the programmed acceleration to the axis phase adjust acceleration (A-0-0154).
5. Get the slave position corresponding to the current master position according to the cam table equation from parameter A-0-0163.
6. Make a single-axis absolute move to the position in A-0-0163 using the AXIS/MOVE command. The drive currently supports only the shortest path mode. If the same direction is desired, do two successive moves.
7. Enable motion using the AXIS/START instruction (GO icon).
8. Wait for the move to be completed by using the AXIS/WAIT instruction or by testing the in-position or alignment status of each drive being aligned.
9. Switch the axis to cam mode using the ELS/MODE instruction.

Aligned Status Bit (Axis Status Register bit 8)

This bit provides the status of cam alignment. It is set to (1) if the axis is aligned to the cam, and (0) if it is not aligned.

The following conditions set this bit to (0):

1. The axis is not configured in the program to be a cam axis.
2. A valid cam is not active for this axis.
3. The absolute value of (position of the axis - slave position from cam equation) is greater than the in-position window (drive parameter S-0-0057).

The following conditions set this bit to (1):

1. The axis is synchronized to the master
2. The absolute value of (position of the axis - slave position from cam equation) is less than or equal to the in-position window (drive parameter S-0-0057).

C.8 Drive Cams

Indramat digital drives include the ability to store and execute cams with up to 1024 points The positions are defined with a normalized table that is based on the ELS master position. The slave axis units and scaling are defined by the shaft distance (H). It is possible for the slave axis to have either rotary or linear units.

Drive Cam Equation

The electronic cam mode on the digital drives can be described by the following equations:

a) When switching to synchronization mode, the following equation is immediately applied in the drive:

Scmd = (H * Cam(Mcmd - Mph)) + Sph

b) As long as synchronization mode is active, the drive updates the position difference:

ΔScmd = (H * ΔCam(Mcmd - Mph)) + ΔSph

Mcmd	Position command of the master (degrees)	C-0-0157
Mph	Master phase adjust (degrees)	P-0-0061
Cam()	Normalized 1024 point cam table	P-0-0072, P-0-0092
H	Cam shaft distance (deg, mm, or in.)	P-0-0093
Sph	Slave phase adjust (deg, mm, or in.)	A-0-0151, P-0-0034
Scmd	Commanded position to slave	S-0-0047

The active cam is switched after the parameter stored in P-0-0094. A status of the active cam is in P-0-0089.

Axis Configuration

Drives that use the DDS cams are configured as ELS axes with the Cam Synchronization mode. This is a fourth ELS mode in addition to velocity synchronization, phase synchronization, and CLC cam mode. Note that the drives are set up at initialization time to use either CLC cams or drive-resident cams. An axis may not switch between both types of cams or switch to cams not stored on that drive.

Drive Cam Table Parameters

For drive-resident cams, the drive parameters P-0-0072 and P-0-0092 are used. The cams on the drive always contain 1024 points. These points are distributed in equal distances over a range of 360 degrees, resulting in a point for every .35 degrees of the master. The table values have a permissible range of -200% to +200%.

For example:

Point Number	Value (Percent)
1	0.000000
2	1.100000
3	2.300000
….	….
1023	0.400000
1024	0.000000

The table is made up of points weighted in percent of the H factor (or shaft distance). There is a linear interpolation between the points dependent upon master axis position. The position command is determined by multiplying the table values by the H factor, which can be changed while the cam is running.

Drive Cam Alignment

It is possible to align a drive-based cam slave axis to the output position of the cam by using the align status, the current slave position, and single-axis positioning.

The following procedures can be used in the CLC user program before switching a cam axis into synchronization mode. *As an option to this, Dynamic synchronization can also be used.*

If modulo positioning is enabled:

1. Check the Phase Adjusted bit (axis status bit 4) to determine if a move needs to be performed.

2. Switch the axis to single axis mode using the ELS/MODE instruction.

3. Set the programmed velocity. A good value to use is the homing velocity (S-0-0041).

4. Set the programmed acceleration. A good value to use is the homing acceleration (S-0-0042).

5. If shortest path motion is needed, set drive parameter P-0-0013 to 0. For motion in the positive direction only, set this parameter to a 1.

6. Parameter P-0-0034 (Actual value of additive pos. cmd.) contains the difference between the current axis position and the position at the output of the cam generator corresponding to the current master position. Read this parameter into a variable.

7. Calculate the distance for an incremental move to: distance = S-0-0103 - ((P-0-0034 - slave_offset) % S-0-0103), where slave_offset is the slave phase offset to be established at synchronization.

8. Start a single-axis incremental move of distance in the equation above using the AXIS/MOVE command.

9. Enable motion using the AXIS/START instruction (GO icon).

10. Wait for the move to be completed by using the AXIS/WAIT instruction or by testing the in-position status of each drive being aligned.

11. Switch the axis to cam mode using the ELS/MODE instruction.

If absolute positioning is enabled:

1. Check the Phase Adjusted bit (axis status bit 4) to determine if a move needs to be performed.
2. Switch the axis to single axis mode using the ELS/MODE instruction.
3. Set the programmed velocity. A good value to use is the homing velocity (S-0-0041).
4. Set the programmed acceleration. A good value to use is the homing acceleration (S-0-0042).
5. Parameter P-0-0034 (Actual value of additive pos. cmd.) contains the difference between the current axis position and the position at the output of the cam generator corresponding to the current master position. Read this parameter into a variable.
6. Calculate the distance for an incremental move to: distance = -(P-0-0034 - slave_offset), where slave_offset is the slave phase offset to be established at synchronization.
7. Start a single-axis incremental move of distance in the equation above using the AXIS/MOVE command.
8. Enable motion using the AXIS/START instruction (GO icon).
9. Wait for the move to be completed by using the AXIS/WAIT instruction or by testing the in-position status of each drive being aligned.
10. Switch the axis to cam mode using the ELS/MODE instruction.

Phase Adjusted Status (Axis Status Register bit 4)

This bit provides the status of cam alignment, as well as the status of a phase adjust. It is set to (1) if the axis is aligned to the cam, and (0) if it is not aligned. It is set from the cyclic value of drive parameter S-0-0182, bit 8. Note that the 'phase adjusted' bit is set before cam synchronization only if phase offset is performed on the drive, not the CLC.

The following condition sets this bit to (0):

The absolute value of (position of the axis - slave position from cam equation) is greater than the position synchronization window (drive parameter S-0-0228).

The following condition sets this bit to (1):

The absolute value of (position of the axis - slave position from cam equation) is less than or equal to the position synchronization window (drive parameter S-0-0228).

Drive Cam Configuration Notes

1. Cam shaft distance (H) is controlled through the service channel by default. In the cam/engage command, it is only sent when it changes. To put cam shaft distance in cyclic data, set parameter A-0-0180 or 0181 to (32768+93). The cam/engage command will then send the cyclic value. Note that the drive interpolates the H factor so that there are no discontinuities in the profile.

2. The slave phase offset for drive based cams is the same parameter used in ELS (A-0-0151 and S-0-0048). By default, this parameter is in the cyclic data.

3. The master phase offset for drive-based cams is sent through the service channel (P-0-0061) by default. Like the H factor, cyclic data can be used by setting A-0-0180 or A-0-0181 to (32768+61). Note that unlike the slave phase offset or the H factor, there is no filter on this value, on either the drive or the CLC.

4. For all cam axes, the positioning is set to rotary in the user program by default. To run a cam with a linear axis, add a parameter init. of bit 2 in A-0-0004 to 0 after the els_init command.

5. to 0035 are not used for drive-based cam axes. These are battery-backed parameters on the CLC that are used only for CLC-based cams.

6. to 0163 are not used drive-based cams. The cyclic slave phase offset parameters (A-0-0151 to A-0-0157) are inconsistent with CLC cams but are consistent with ELS slave axes.

7. If parameter A-0-0164, bit 6 is set to (1), the phase offset can be initialized to any value before the drive is switched into synchronization mode. This allows the user to specify a phase offset according to the range of motion. If the bit is set to (0), the CLC automatically establishes relative synchronization by setting the phase offset to the value in P-0-0034, as it does in ELS phase synchronous mode.

8. Parameter A-0-0163 is used for CLC-based cams only. Its equivalent for drive-based cams is: slave_position = S-0-0051 - P-0-0034.

9. Using the cam/build instruction, it can take up to 6 seconds to calculate the cam and send it to the drive.

C.9 Differences between CLC and Drive Cams

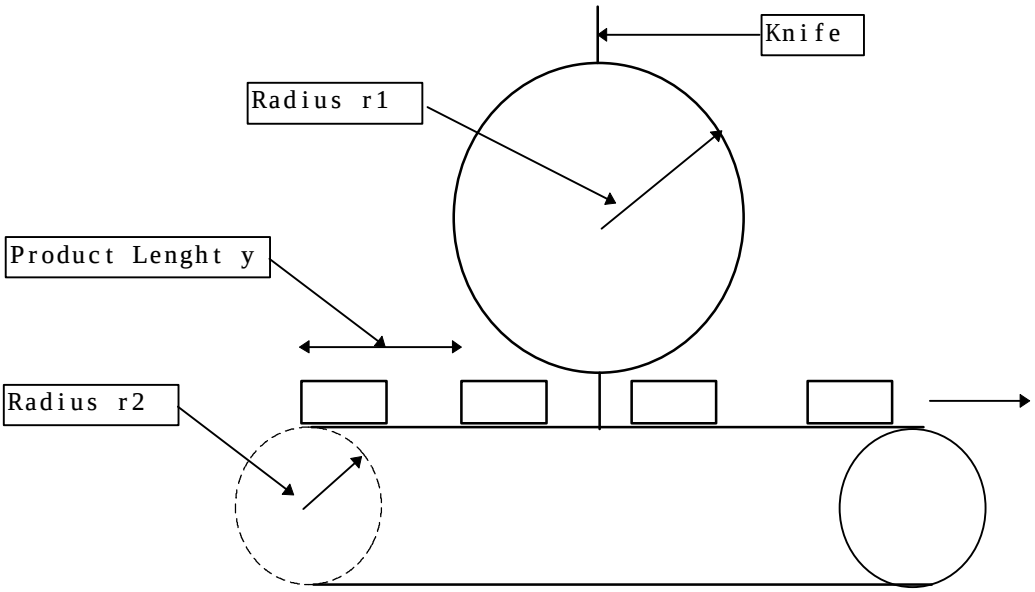
In most cases, drive-resident cams should be used if the drive software supports them. There are some features not implemented in the drive at this time that may require the use of cams stored on the CLC.

	CLC Resident	Drive Resident
Drive Versions Supported	All SERCOS drives	Indramat DDS2.2 and similar drives (DIAX03)
Number of cam axes allowed	4	maximum allowed in system
Number of cams allowed	8	2 * number of drives in system
CLC resources required	large amount	minimal
Cam Equation	Scmd = H*CAM[(M/N)*Mcmd + Mph] + L*Mcmd + Sph	Scmd = H*CAM[Mcmd] / 100 + Sph
Cam Factors	M, N, H, and L	H
Table Format	2 column: x= master, y=slave	1 column of slave values
Master Intervals in Table	variable through x-column	fixed at 0.35 degrees
Number of Points in Table	200-1024	always 1024
Slave Units	Position Units	Percent of H factor (100.0 is 1 unit)
Table Switching Point	0 degrees	selected by parameter
Alignment before Sync.	Absolute move to A-0-0163	Incremental move of -(P-0-0034 - initial offset)
Interpolation update rate	2ms with 2 cams, 4ms if > 2 cams	500 microseconds

C.10 Application (Rotary Cutoff)

In this application a cam profile is designed that drives a two bladed knife used to seal and cut pre-wrapped candy bars on a conveyor belt.

Consider the following drawing:



Rotating Knife Application

In this application the ELS line master is driving motor r2 directly and motor r1 is the cam axis.

A cam table with a dwell is need so that during the dwell the knife will match line speed to allow for proper sealing and cutting. The placement of the dwell within the cam will effect the master phase adjust, Mph, needed for centering the cam to the product cycle.

Define Ry to be the number of master revolutions/product cycle.

Then,

$Ry = y/(2*\pi*r2)$

To set the cam cycle to the product cycle use M and N.

One cycle of the normalized cam for every product implies:

Ry*(M/N) = 1 revolution

so,

N = Ry

M = 1

For the moment, assume H = 0.

L is set to match distance and speed of the knife during a cycle to that of the product.

The equation for L is as follows:

(2*π*r1)*(Ry)*L = y

L = r2/r1

The distance factor (2*π*r1)*(Ry) is the total distance traveled by the knife in one product cycle. L is used to scale this distance to y, the product length.

Define B to be the number of knife blades, thus B=2, and d to be the distance between knife blade(s).

If y equals d then the distance traveled by the knife in one product cycle is the same as the product length. In this case no cam is need and H = 0.

If y is not equal d then the knife never hits the product at the right time in the cycle. This is where H comes in and a cam is need.

H is used to scale the cam to make up for any length difference between y and d.

The distance between knife blades is defined by:

d =(2*π*r1)/B

The distance adjustment needed each cycle is given by:

d_adj = d - y

The cam is normalized so it makes one cycle per product cycle, therefore:

B*d*H = d_adj

H = (d - y)/(B*d)

H = (1 - y/d)/B

Note that B*d = 2*π*r1.

H may be zero. When H is zero the cam profile is not needed and this application can be done without a cam.

H may be negative. This happens when the distance between the blades is smaller than the product length, so that the knife will slow down during the cam cycle then speed up and be at line speed during the dwell. The distance adjustment attributed to the cam is exactly d-y.

H may be too negative. In this case the cam will slow down the knife so much that it actually will reverse direction and travel backwards. This is not acceptable and only happens when the product length is at least twice that of the knife. For this case, L must be set to zero and a cam profile developed that will start from rest, accelerate to match line speed then decelerate to a stop. The details of this approach is not addressed here.

H may be positive. In this case the distance between the knife blades is larger than a candy bar so the knife will speed up during the cam cycle then slow down and match line speed during the dwell.

H may be too positive. In this case the amount of distance the cam profile must make up is so large and the line speed may be so high that the acceleration and deceleration rates are not acceptable to the motors.

Phase can be added or subtracted to center the cutting edge of the knife exactly between the candy bars.

For pre-production setup initialize the ELS master and master phase adjust to zero degrees. Center the cam dwell at zero degrees of the cam. Set the slave phase adjust so that the knife is positioned at its first cut.

During production, use the master phase adjust for registration correction as needed.

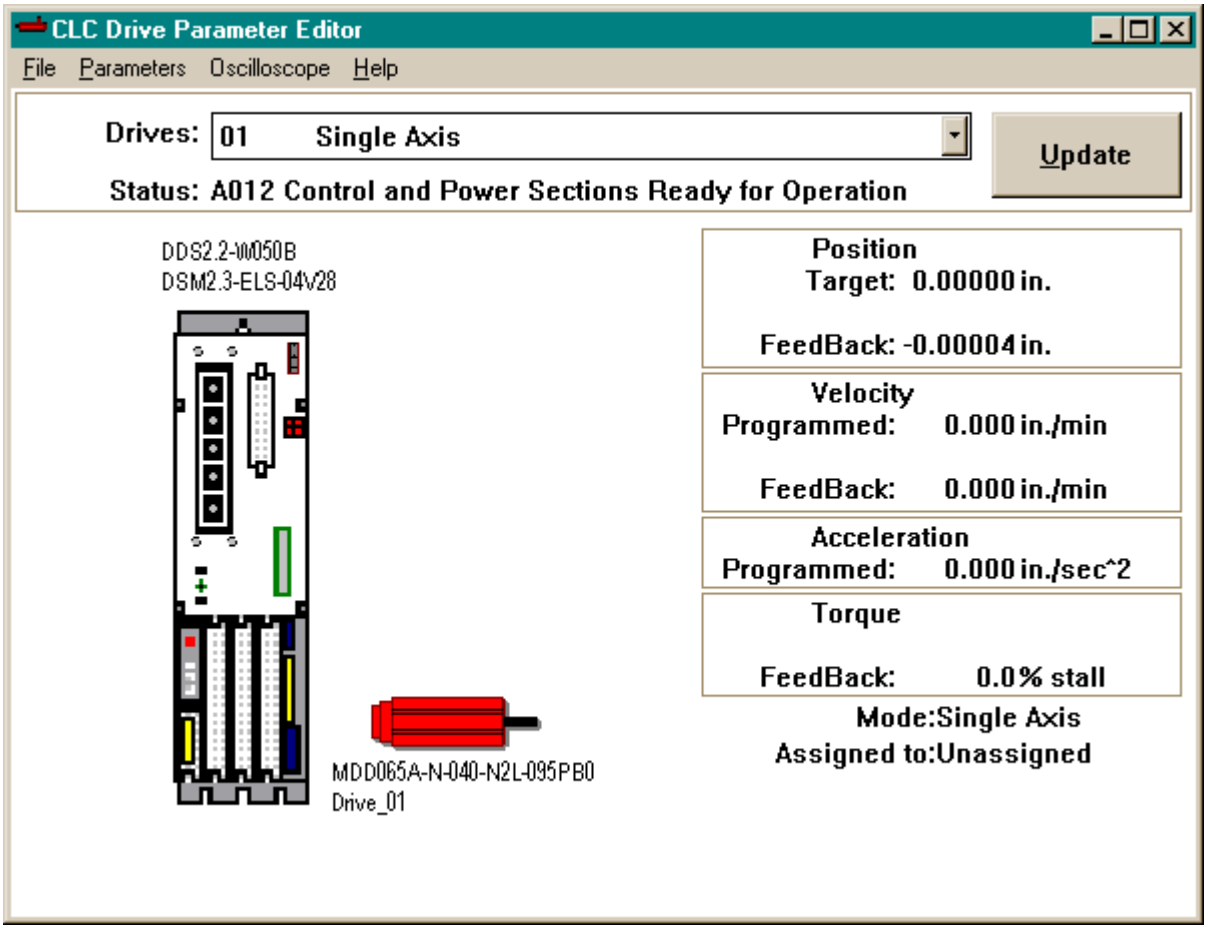
D CLC Drive Parameter Editor

D.1 Overview

When opened, the CLC Drive Parameter Editor uploads the current status information for drive #1 (the default). The current drive status is also displayed. The position, Velocity and Acceleration values from the CLC / TRANS 01-D card are displayed along with the feedback status from the selected drive.

The Drive # box allows selection of another drive by entering a drive number or scrolling with the up/down list buttons. Clicking Update uploads the current values for the selected drive.

Since the drive internally generates rate profiles for single axis motion, the programmed acceleration is also displayed. Acceleration is not shown for coordinated motion since the CLC / TRANS 01-D path planner manages acceleration for coordinated motion.

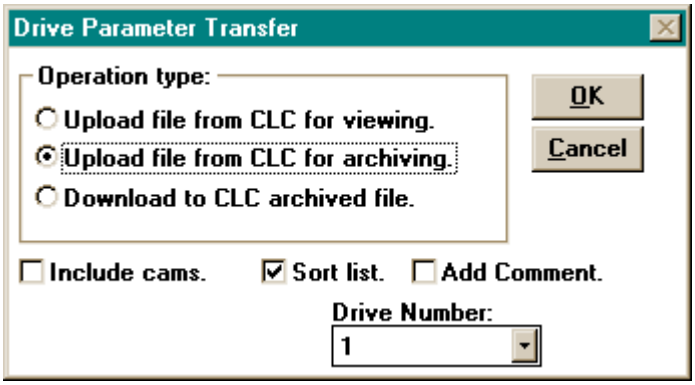


D.2 File Menu

Transfer Parameters

This menu item uploads parameters to a file for archiving or viewing, and downloads archived parameters from a file to a CLC / TRANS 01-D card.

Note: This option only transfers the parameters for the servo drive; it does not transfer any other TRANS 01-D parameters.



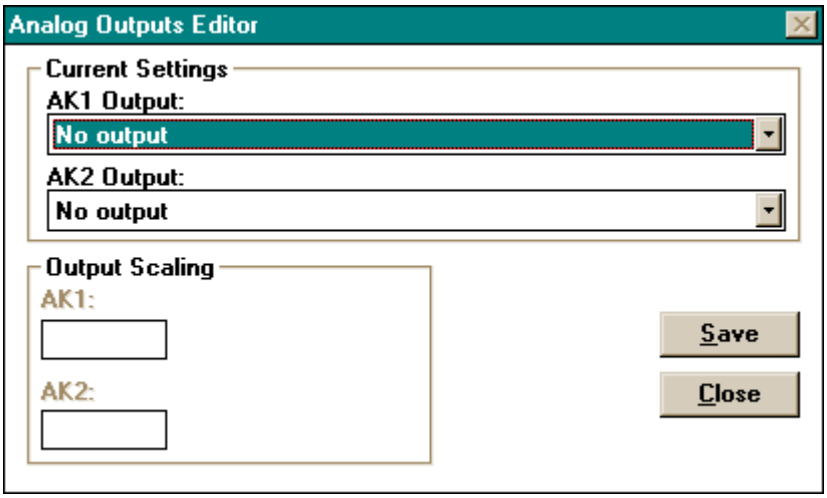
Each drive has one parameter set that can be transferred. Drive parameters need a drive number. The parameter set for the selected drive connected to the CLC / TRANS 01-D may be transferred through the SERCOS communication system.

Parameters may be uploaded in one of two formats. Uploading for archiving saves the file with the ".prm" file extension in the :\\SAVESET directory, with the data in the proper format for downloading to the CLC / TRANS 01-D. Uploading for viewing saves the file to the same sub-directory as a text file with a ".txt" extension and may be viewed using Notepad or another ASCII text editor or file viewer. A *.txt parameter set uploaded for viewing cannot be downloaded to the CLC / TRANS 01-D.

D.3 Parameter Menu

Analog Outputs

Choosing Analog Outputs from the Parameters menu opens the Drive "n" Analog Output Settings dialog box. This is used to manage signal selection and scaling of the AK1 and AK2 analog output drive signals. By default, the current values for drive #1 are uploaded and displayed.



The Drive # box allows selection of another drive by entering a drive number or scrolling with the up/down list buttons. Clicking Update uploads the current values for the selected drive. Scrolling lists of signals available for both outputs include:

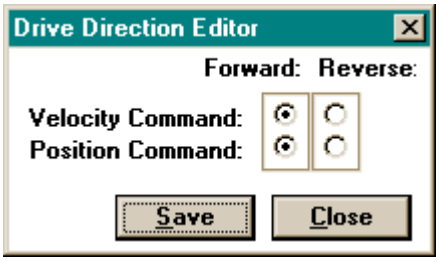
- No output
- Current command value
- Velocity feedback value
- Velocity command value
- Position command value difference between each SERCOS cycle (2ms).
- Position feedback value 1 (motor encoder)
- Position feedback value 2 (external encoder)
- Lag error
- Sine signal from motor encoder
- Cosine signal from motor encoder
- Sine signal from external encoder
- Cosine signal from external encoder

Individual data entry boxes allow independent scaling of each output (maximum 10 volt output) to meet the requirements of an external indicator (analog or digital panel meter, etc.)

Clicking on the Save button downloads the signal selections and scaling factors to the selected drive through the CLC / TRANS 01-D and SERCOS communications.

Drive Direction

Drive Direction is an edit window for viewing and setting a drive direction parameters. These parameters invert the direction of the commands to the drive. (i.e. A 5 inch move will move 5 inches negative if the Position Command is set to the reverse direction)

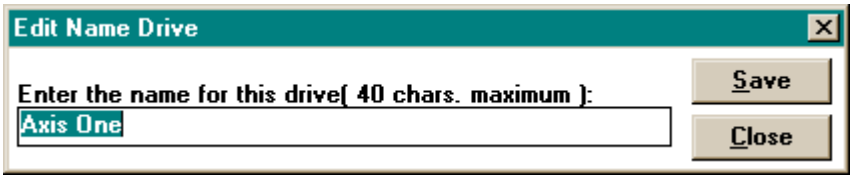


To view settings for a drive, select the drive number in the main window and press the Update button.

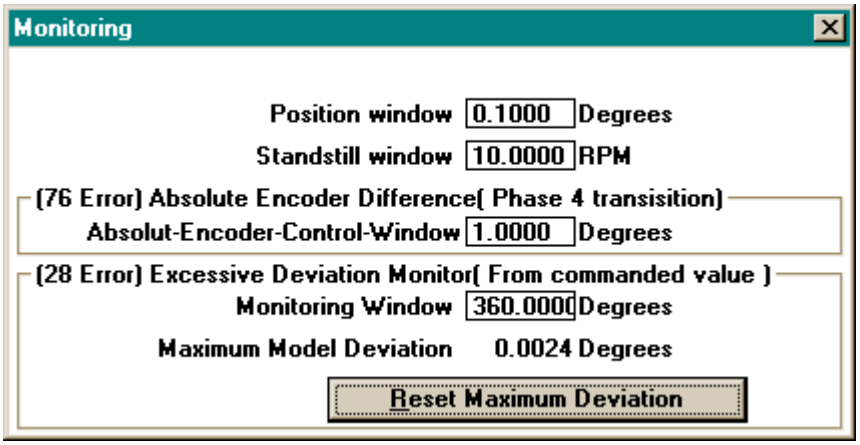
To change a drives settings, switch to parameter mode, then select the direction buttons and press the Save button. ***See drive parameters S-0-0043 and S-0-0055***.

Drive Name

This window displays the name of the currently active drive. The default name is the axis number however a custom name can be assigned and saved to any of the drives on the SERCOS ring.



Drive Monitoring



Selecting Drive Monitoring from the Parameters menu opens the Drive Monitoring window and refreshes the following drive information:

Position window (S-0-0057) sets the tolerance distance used to determine if it's in position.

Standstill window (S-0-0124) sets the velocity to determine if it's at a standstill.

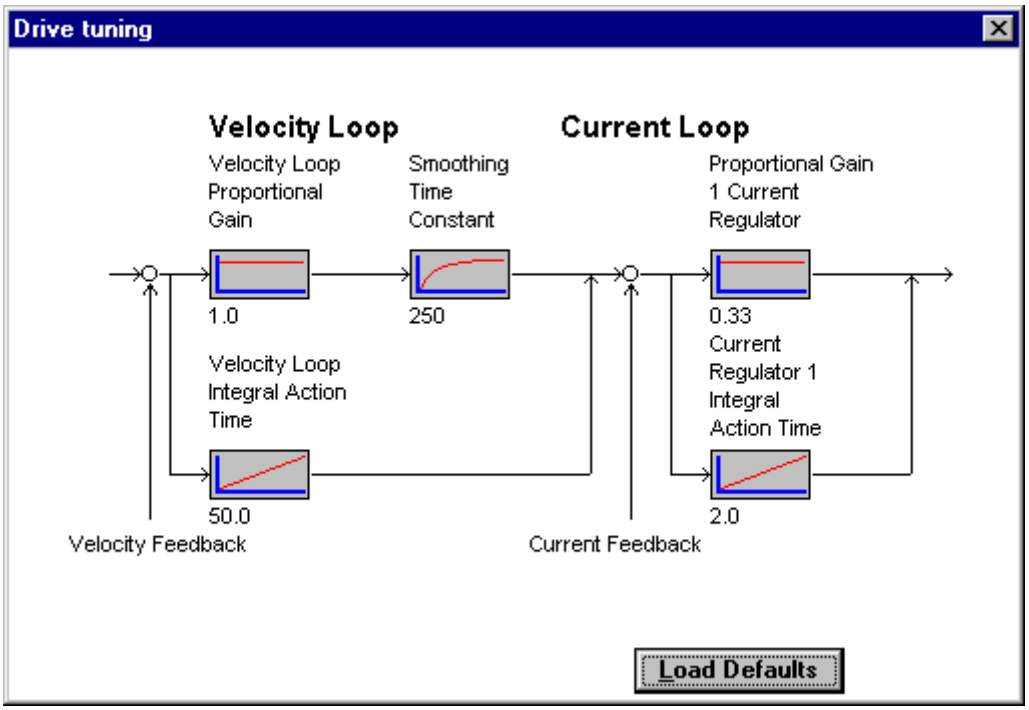
Absolute Encoder Difference (P-0-0097) sets the maximum distance the motor can move when off, without causing an error 76 on powerup to phase 4.

Monitoring window (S-0-0159) sets the maximum position unit (or percent for some drives) from the command value before the drive issues an error 28.

The current **Maximum Model Deviation** from the command value is displayed in position units (or a percent for some drives). This value can be reset using the button on the bottom of the screen.

Drive Tuning

Selecting Drive Tuning from the Parameters menu opens the Drive Tuning dialog box for the currently active drive. The adjustments within the Velocity loop are related to a machines performance. The Current Loop adjustments are set according to the respective motor/drive combinations and should not be altered from their initial operative setting.



The Velocity Loop **Proportional Gain** data entry box adjusts the gain of the loop feedback path (SERCOS parameter S-0100). The gain is initially adjusted by the drive/motor combination for a 1:1 load/motor inertia ratio. This parameter value may be set from 0 to 65535

The **Smoothing Time Constant** data entry box sets a low-pass filter that limits the bandwidth of the feedback loop and reduces digital quantization effects (Indramat parameter P-0-0004). The time constant is set in microseconds, any entry under 250µs switches off filtering.

The Velocity Loop **Integral Action Time** data entry box also sets a low pass filter time constant integrating the velocity loop feedback signal (SERCOS parameter S-0-0101). This parameter is typically used to adjust the loop response time, matching the load to motor and reducing overshoot that may result from a rapid (step) change. This filter has a lower frequency breakpoint than the Smoothing/Roll-off value.

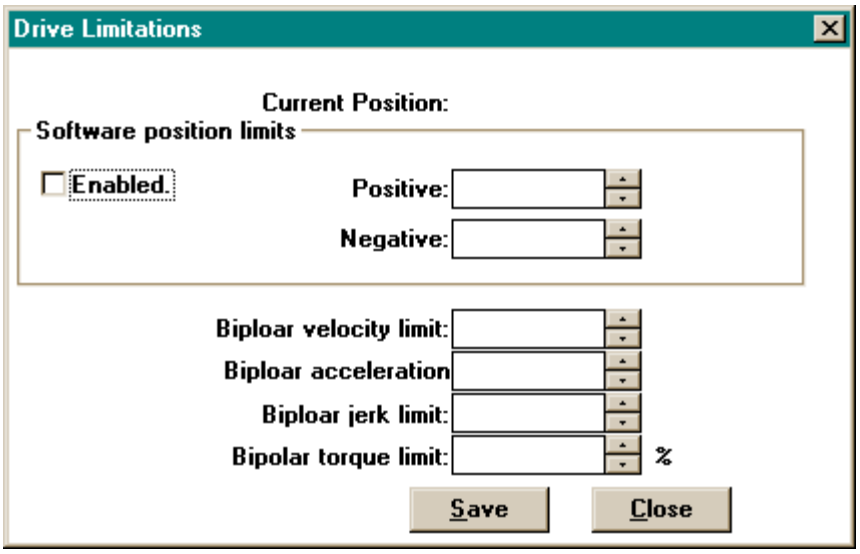
The Current Regulator **Proportional Gain** (S-0-0106) and **Integral Action Time** (S-0-0107) adjustments are used for the initial tuning of respective motor/drive combinations. This adjustment should not be changed once it is set for a specific system.

Clicking the Defaults button loads default parameter values into the dialog box data entry fields (CLC / TRANS 01-D must be in Parameter Mode). The default values assume a 1:1 ratio of load inertia to motor inertia.

For further information on tuning Indramat drives and tuning parameters ***refer to the Digital Servo Drive User's Manual.***

Drive Limits

Positive and Negative **Travel Limits** set floating point values for the drive Positive (S-0-0049) and Negative Position Limit Value (S-0-0050) parameters. The Travel Limits are not affected unless Enabled is checked.



The **bipolar velocity limit** value (S-0-0091) determines the maximum allowable speed in either direction. If the velocity limit value is exceeded, the drive responds by setting the message "ncommand > nlimit" in Class 3 Diagnostics (IDN S-0-0013).

The **bipolar acceleration** parameter (S-0-0138) reduces the maximum acceleration ability of the drive symmetrically around 0, to the programmed value in both directions.

The **bipolar jerk limit** value (S-0-0349) determines the maximum allowable jerk in either direction.

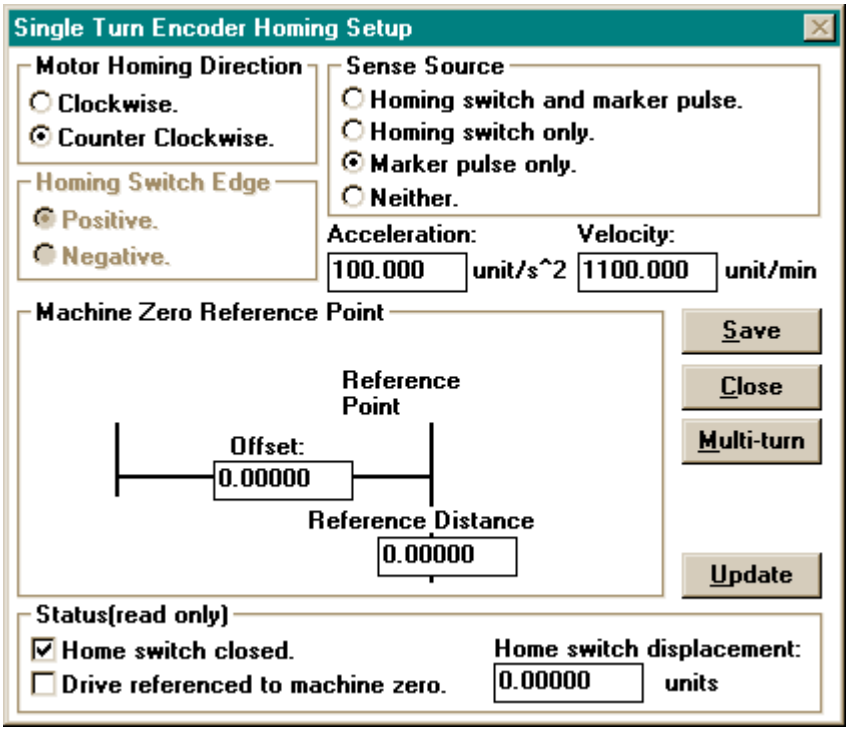
The **bipolar torque limit** value (S-0-0092) determines the maximum allowable torque in either direction. If the torque limit value is exceeded, the drive sets the message "T > Tlimit" in Class 3 Diagnostics (S-0-0013).

Drive Reference

This menu item will automatically sense the active drives motor encoder type and launch either the single or multi-turn encoder dialog box.

Single-turn Encoder VisualMotion:

Selecting incremental encoder from the drive reference menu opens the Homing Setup dialog box and uploads the current parameter values for drive #1 (the default).



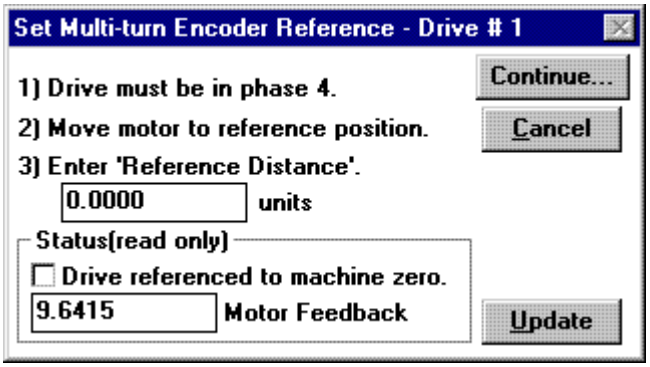
Homing does not need to be configured if the Home icon or AXIS/HOME text language command are not used in user programs. Homing is an internal capability of Indramat intelligent drives and requires only that the CLC / TRANS 01-D send a home command to the drive. The actual homing procedure performed by the drive is set by the drive parameters.

Clicking Update uploads the current values for the selected drive.

Because the setup of drive homing parameters is highly system dependent, ***refer to the Drive User's Manual*** for information on the various homing routines and the requirements for setting up the homing parameters.

Multi-turn Encoder

Reference Absolute Encoder instructs the DDS drive to use the current shaft position as the position indicated in the "Reference Distance".



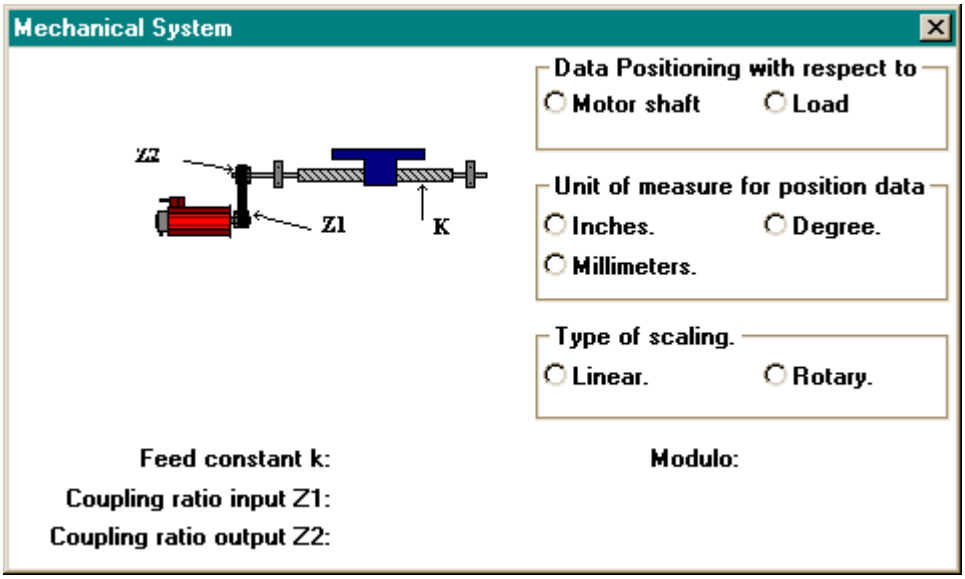
Set the drive number to the drive to be referenced. Move the motor to the position to be referenced. Set the "Reference Distance" to the value to be assigned to this position.

Press the Continue button. A second screen will inform you that all motion on this drive will be referenced to these values. Press OK to accept, or Cancel to Abort.

After successful completion of this procedure, the checkbox 'Drive referenced to machine zero' will be checked.

Mechanical

Selecting Mechanical from the Parameters menu opens the Mechanical Setup Drive dialog box and uploads the current values from the drive. This dialog box allows easy access to several important parameters, which must be set before running any motion programs.



Data Positioning with respect to the **load** configures the drive to use Encoder 2 to close the position loop and provide cyclic feedback from drive parameter S-0-0053. This sets *S-0-0004 - Bit 11: Use Secondary Encoder for Positioning* equal to 1. This option must be set if the CLC / TRANS 01-D is used with DDS-2.1 linear motor firmware.

The **Units of measure for position data** selection has a choice of inches, millimeters or radians for the system-wide unit of measurement (A-0-0005).

The **Type of scaling** selection can be linear or rotary (A-0-0004 bit 2). When linear is selected absolute positioning is enabled in the drive. When rotary is selected, position is in degrees, velocity in RPM, and acceleration in radians/sec2.

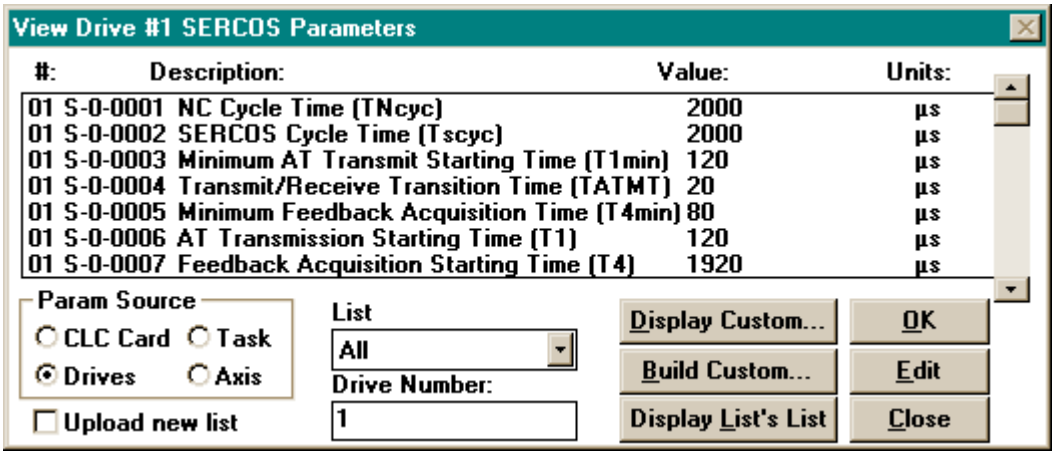
The **Feed Constant** allows setting the ratio of movement in system units resulting from each revolution of the driven shaft (S-0-00123). For example, a 5 TPI ball screw provides 0.200 inch movement per revolution. The **Calculator icon** opens a scientific calculator that may be used to determine the feed constant.

Output (Z2) and Input (Z1) Revolutions permits setting the ratio between the motor shaft and driven shaft. Integer values permit preservation of maximum system accuracy with ratios that result in repeating decimals (i.e. 1:3 = 0.333333...). These values are set in the drive's Input Revolutions of Load Gear (S-0-0121) and Output Revolutions of Load Gear (S-0-0122) parameters.

The **Modulo** value (S-0-0103) is indicated as a maximum rotational value in which the motor will turn before resetting the position to zero. The default value when operating in modulo mode is 360. (Modulo mode is set in the drive by setting bit 7 of IDN S-0-0076, Scaling Options for Position Data).

Overview

Selecting Overview from the Setup menu opens a View Drive # Sercos Parameters dialog box. This dialog box may be used to view and setup all Drive and CLC / TRANS 01-D user accessible parameters.



Parameters may be uploaded for display or editing by selecting one of the “Param Source” radio buttons, then clicking the OK button. Choosing Drives or Axes requires the entry of a Drive or Axes Number in the enabled data entry box. Choosing CLC / TRANS 01-D card allows uploading the CLC / TRANS 01-D card system parameters. Selecting Task requires choosing of one of the four CLC / TRANS 01-D tasks from the enabled pull-down menu.

After the selected list has been loaded, you may scroll the list by clicking on the list scroll bar's up and down arrowheads, or clicking and dragging the scroll button. Clicking and holding (or dragging) the scroll bar button will display the number of the first viewable parameter in the display list in the upper left of the title bar.

Selecting a list entry and clicking the Edit button opens a parameter editing dialog box that allows you to change the parameter value and update the parameter by downloading it to the CLC / TRANS 01-D or DDS-2 drive.

In addition to uploading all the parameters of a set (i.e., all drive or task parameters, etc.) you may build a custom set of parameters by clicking on the Build Custom... button and opening a Build Custom Display dialog box.

When the Build Custom Display box opens, the parent View ... Parameters dialog box remains active. Selecting a list item from the parent box's list, then clicking Add in the build box adds the selected parameter to the custom parameter list. Only one parameter may be added at one time. Selecting one of the parameters in the custom list, then clicking Delete, removes the item from the custom list.

Since the parent dialog box remains active, the parameter list in the parent dialog box may be changed by re-selecting a Param Source and type. Parameters from the newly uploaded list may then be added to the custom list in the build dialog box. In this manner a custom list may be built by selecting any available parameter from any parameter list.

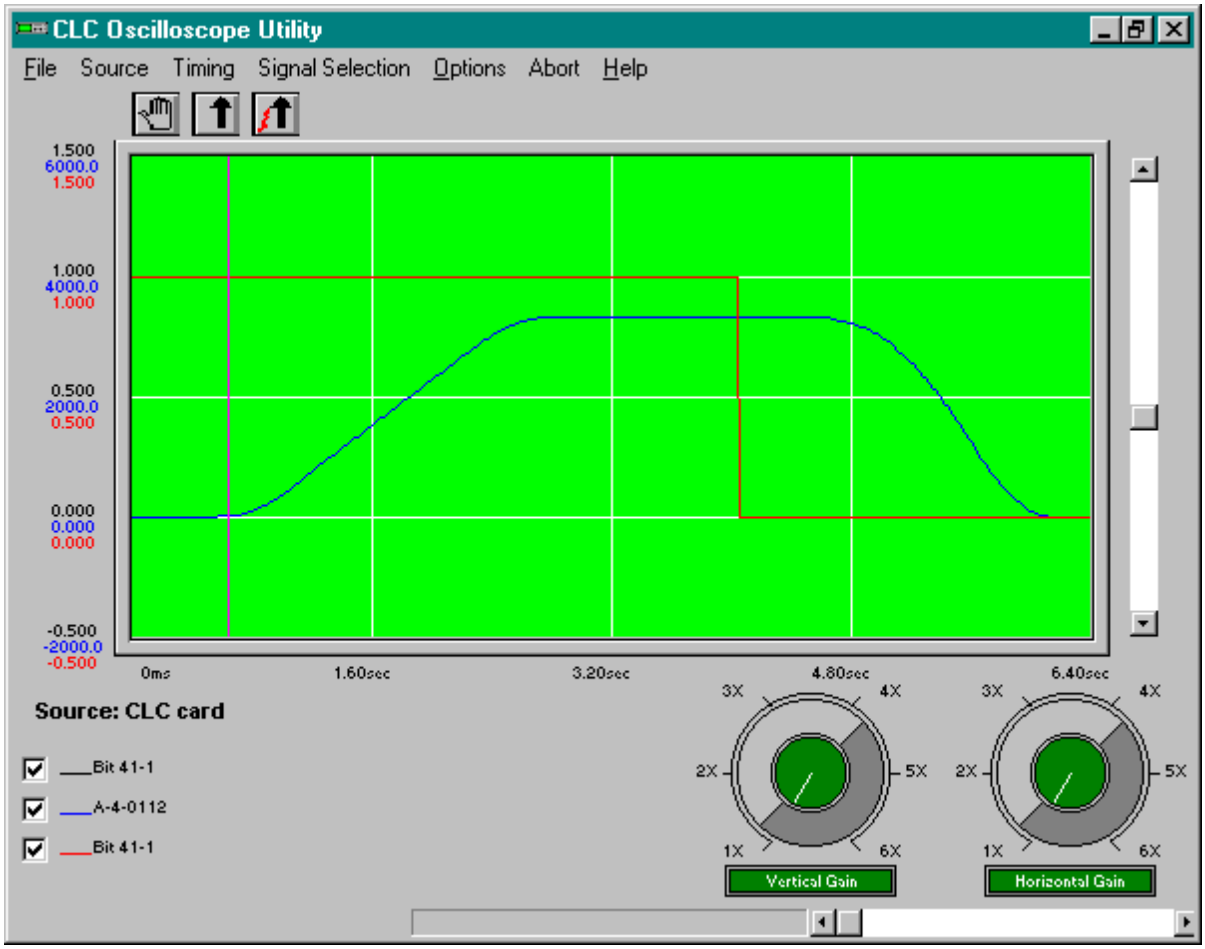
A previously saved custom list may be loaded and used as a source for building another custom list, however, it must be the first list loaded. You cannot load a custom display list once another list has been loaded without closing the Build Custom Display dialog box.

Clicking on Save pops-up a Name Custom Display dialog box permitting entry of an eight character maximum filename. Custom parameter lists are stored in the *CLC.INI* file in the appropriate sub-directory.

Clicking on the Display Custom button opens a Display Custom List dialog box with a list of the currently saved custom parameter lists by name. Selecting a list and clicking on OK or double clicking the list item uploads and displays the parameters in the selected list.

D.4 Oscilloscope

This utility is used to capture and display run-time data. The capture can be on the CLC card or on a drive that supports this feature. Selected data is acquired on the drive or card, passed to Visual Motion, and displayed on a graphical format. The graphical display and supporting data can be printed, or the data can be saved to a file for additional review.



File Menu

Used for getting file data, saving data to a file, printing, and exit.

Open - data from user selected input file is loaded into input data list-box.

Save - data from user selected output is loaded into output data list-box.

Print Output - the oscilloscope graph and its related data table is sent to the printer.

Exit - terminates this utility

Source Menu

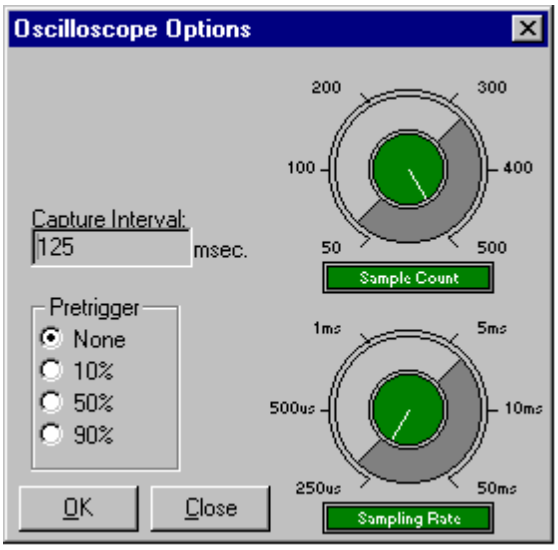
Selects the source from which the oscilloscope will gather signal data.

Drive 1 to n - lists the drives on the ring that support the oscilloscope feature.

Card - when selected the oscilloscope can then read card variables, parameters and

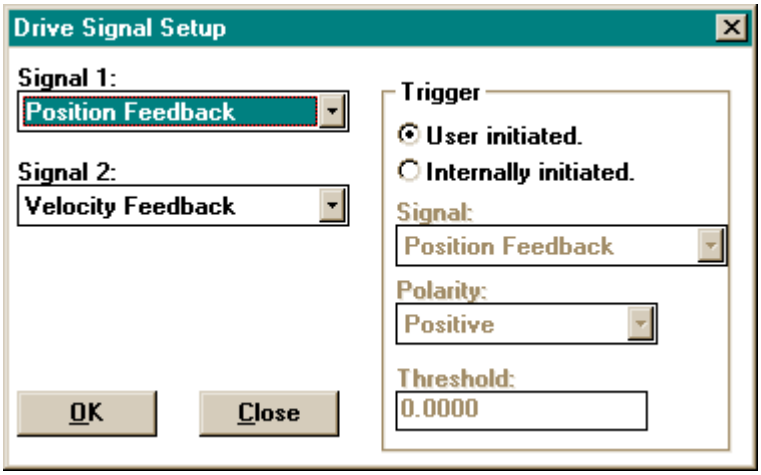
Timing

Chooses the sample count, sample rate, and a 50% pre-trigger. Sample counts are 50, 100, 200, 300, 400, and 500 for both card and drive captures. Sampling rates for the card are 2, 4, 8, 16, 32, and 64ms, for the drive they are 0.25, 0.5, 1, 5, 10, 50ms.



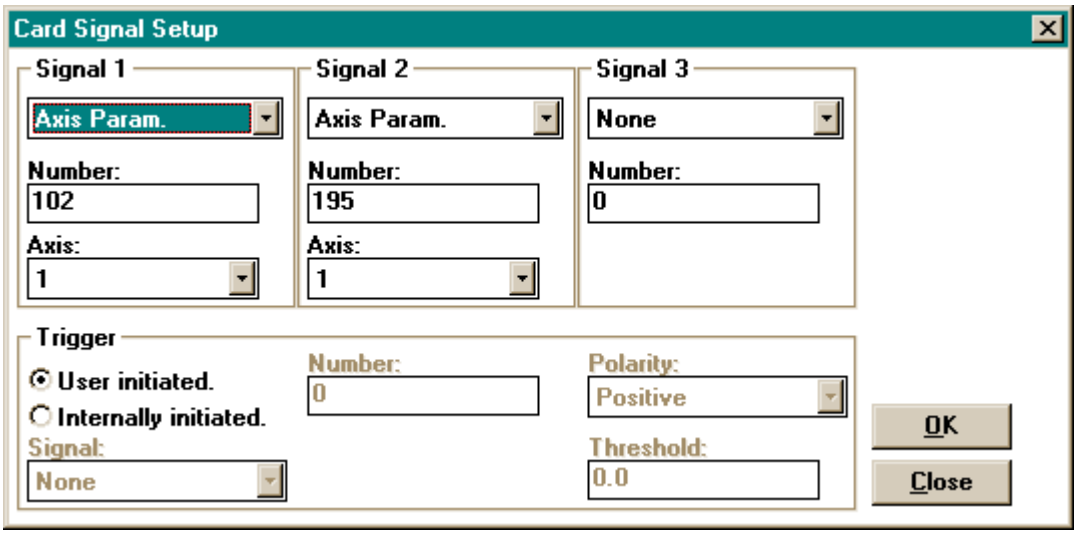
Signal Selection

Selects the signals and trigger options.



When a ***drive*** is the selected source used to gather data the possible signals include:

- Position Feedback
- Velocity Feedback
- Velocity Deviation(from commanded value)
- Position Deviation(from commanded value)
- Torque Command Value(required to maintain the commanded Velocity/Position)



When the CLC ***Card*** is the selected source used to gather data the possible signal include:

- Program Floats(Fx)
- Program Integer(Ix)
- Global Floats(GFx)
- Global Integers(GIx)
- Axis Parameters of drives on SERCOS ring*
- Registers(value)
- Registers(bit)

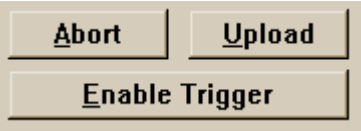
**Axis parameter must be in cyclic telegram, use parameter A-0-0185 and A-0-0195 to add other drive parameters to cyclic data.*

For either source the sample acquisition may be ***user*** or ***signal*** threshold triggered:

For ***user*** triggered captures, data acquisition starts as soon as the necessary handshake is complete. This type of start capture is not deterministic.

For ***threshold*** triggered the trigger signals are the same as the signals to view. The trigger polarity options are on *positive level*, on *negative level*, or on *either*. Signal threshold is the signal level to trigger on.

Abort, Upload and Enable Trigger



The **Enable Trigger** button starts the data acquisition cycle. It sets the trigger condition, waits for data capture, uploads the data, and displays it.

The **Upload** button uploads data already captured and displays it.

The **Abort** button terminates a data acquisition cycle.

I Index

absolute point table....5-22
Absolute Synchronization....C-4
ACAM....C-15, C-18
Accel 6-10
acceleration profiles....1-9, C-5
acceleration/deceleration....5-22
Activating a Program (PA)....A-25
Add Event Function....4-11
Add Subroutine....4-11
addressing
indirect....5-18
Aligned Status Bit....C-23
Analog Outputs D-3
Application (Rotary Cutoff) C-30
Archive 4-4
arrays5-17
Available Program Memory....3-26
Axis 6-11
Axis Fast Jog Velocity 3-25
Axis Feedback Capture (Registration)..3-108
Axis Jog Control Registers....2-17
Axis Large Increment....3-25
Axis Motion Type....3-87
Axis Parameters
Axis Feedback Capture (Registration)3-108
A-0-0170 Feedback Capture Status3-108
A-0-0172 Probe 1 Negative Captured Position....3-109
A-0-0173 Probe 2 Positive Captured Position 3-109
Axis Setup....3-86
A-0-0003 Axis Motion Type 3-87
A-0-0004 Positioning Options....3-88
A-0-0005 Linear Position Units....3-92
A-0-0011 Drive I/O Card 1 Setup ...3-95
A-0-0014 Drive I/O Card 2 Setup ...3-95
A-0-0017 Drive I/O Card 3 Setup ...3-96
A-0-0020 Maximum Velocity....3-96
A-0-0021 Maximum Acceleration....3-96
A-0-0022 Maximum Deceleration....3-97
A-0-0025 Maximum Jog Increment.3-97
A-0-0026 Maximum Jog Velocity 3-97
A-0-0030 Master Axis for Ratio Function 3-98
Axis Status....3-101
A-0-0100 Target Position....3-101
A-0-0101 Commanded Position....3-101
A-0-0102 Feedback Position....3-101
A-0-0110 Programmed Velocity ...3-101
A-0-0111 Commanded Velocity ...3-102
A-0-0112 Feedback Velocity 3-102
A-0-0120 Programmed Acceleration3-102
A-0-0131 SERCOS Control Word3-103
A-0-0132 SERCOS Status Word 3-103
A-0-0140 Single-Axis Status Word3-103
A-0-0141 Torque Mode Commanded Torque 3-104
A-0-0142 Torque Feedback (cyclic)3-104
A-0-0145 Current Motion Type....3-104
Electronic Line Shaft 3-104
A-0-0151 ELS Programmed Phase Offset 3-104
A-0-0153 ELS Phase Adjust Velocity3-105
A-0-0157 ELS Commanded Phase Offset 3-105
Axis Parameters:....3-86
Axis Setup....3-86
Axis Slow Jog Velocity....3-25
Axis Small Increment....3-25
Axis Status....3-101
Axis Status Registers 2-23
AXIS/EVENT 7-6
AXIS/HOME 7-7
AXIS/INITIALIZE 7-8
AXIS/MOVE (Single Axis, Non-Coordinated)7-10
AXIS/RATIO 7-11
AXIS/SPINDLE (Continuous Velocity Mode)7-13
AXIS/START 7-15
AXIS/STOP 7-16
AXIS/WAIT (Axis Wait For In-Position) . 7-17
AxisEvt....6-16
Backspaces and White spaces A-4
bipolar velocity limit D-7
Bit Labels....4-14
blend segments 5-21
Branch 6-17
Breakpoint Control 4-26
BTC06 1-5
BTC06 Teach Pendant....3-44
Building Cam Tables....C-10
Bus Arbitration Mode 3-37
Bus Release Mode....3-37
Calc 6-19
Call 7-18

Cam
Number 6-24
Cam Alignment C-22
Cam Applications C-10
CAM Builder 4-26
Cam Commands C-10
Cam Equation C-17
Cam synchronization 1-9, C-5
CAM/ACTIVATE 7-19
CAM/ADJUST 7-20
CAM/BUILD 7-21
CAM/STATUS 7-22, 7-23
Camming 6-22
CAPTURE/ENABLE 7-24
CAPTURE/SETUP 7-25
Card Selection 4-17
Checksum A-3
Circle 6-31
CLC
Operating System 1-6
Overview 1-4
System Architecture 1-4
CLC Compiler Base Code 8-2
CLC Cycle Control Considerations 2-13
CLC Input/Output Systems
General Description 2-1
CLC Instructions 7-3
CLC Registers 2-7
CLC Reserved Register Tables 2-31
Clear Current Task 4-10
command execution 5-1
Commanded Position 3-101
Commanded Velocity 3-102
Communication Errors A-2
Communication Protocol A-1
Configuration 4-17
Connecting Icons 6-3
constants 5-13
Conversion functions 5-16
Coordinated Axis 3-87
Coordinated Jogging 2-14
Coordinated Motion 3-74, 3-93, 4-19
Coordinated X Axis 3-75
Coordinated Y Axis 3-75
Coordinated Z Axis 3-75
Current Instruction 3-82
Current Instruction Pointer 3-82
Current Motion Type 3-104
Current X Position 3-80
Current Y Position 3-81
Current Z Position 3-81
Cyclic Data 3-89
Daisy-Chained *6-34*
data format errors 6-82
Data Menu 4-36
data type
type conversion 6-82
data types
conversion 5-15
floating point 5-13
integers 5-13
DATA/SIZE (Configure Memory Allocations) 7-26
DDE 1-5
DDE client interfaces B-1
DDE request B-1
DDEInitiat *B-4*
DDEInitiate *B-5*
DDEPoke *B-3*
DDERequest *B-3*
DDS Setup 3-113
Decel 6-33
DEFINE 7-2
DEFINE (Define Name For Variable) 7-28
DELAY (Suspend Task Execution) 7-29
Diagnostic Code 2-14
Diagnostic Log 4-69
Directives 7-1
DEFINE 7-2
EQU (Equate) 7-1
EVENT/START & EVENT/END 7-2
TASK/START & TASK/END 7-3
Download Block Size A-22
Drive and CLC Parameters and Subclasses A-9
Drive Cam Alignment C-26
Drive Cam Equation C-24
Drive Cams C-24
Drive Direction D-4
Drive Disable Method 3-91
Drive Homing D-8
Drive I/O Card 1 Setup 3-95
Drive I/O Card 2 Setup 3-95
Drive I/O Card 3 Setup 3-96
Drive Limits D-7
Drive Monitoring D-5
Drive Name D-4
Drive Parameter
Drive Status 3-113
Required DDS Setup 3-113
Drive Parameter Editor D-1
Drive Parameters 3-111

Drive Reference....D-8
Drive Status....3-113
Drive Tuning....D-6
Drives.... 4-18, 4-69
Drives on Ring....4-69
Dynamic synchronization....C-5
Edit Menu
Labels
User Labels....4-12
Electronic Line Shaft.... 3-33, 3-104
ELS....6-34
ELS Commanded Phase Offset....3-105
ELS Master Axis....3-33
ELS Master Current Position....3-35
ELS Master Current Velocity....3-35
ELS Master Task Assignment....3-33
ELS Master Type....3-33
ELS Masters....C-1
ELS Phase Adjust Velocity....3-105
ELS Programmed Phase Offset....3-104
ELS Secondary Mode....3-90
ELS Slave....3-87
ELS Slaves....C-2
ELS Synchronization Mode....3-89
ELS/ADJUST (Adjust ELS Axis)....7-30
ELS/DECEL (Set Master Decel)....7-31
ELS/INIT (Initialize ELS Axes)....7-32
ELS/MODE (Set ELS Axis Mode)....7-34
ELS/STOP....7-35
ELSAdj....6-37
ELSMode....6-38
End of Message....A-3
EQU (Equate)....7-1
Erase All Forcing Masks (RE)....A-35
Erasing (Deleting) a Program (PE)....A-24
Error Reaction Mode....3-17
Event....6-39
Event Functions....4-16
Event Table
Event Table Data....A-20
Event Table Data, Row Format....A-21
Event Tables....5-10, A-20
EVENT/DONE (Signal Event Completed)7-36
EVENT/END (Mark End of Event)....7-38
EVENT/START & EVENT/END....7-2
EVENT/START (Start of Event function)7-39
EVENT/TRIGGER (Trigger a Task Event)7-40
EVENT/WAIT (Pause Task for Event Done Signal)....7-42
events.... 1-6, 4-36, 5-2
Distance based.... 5-3
feedback capture.... 5-9
interrupt input events.... 5-6
repeating axis position.... 5-5
Time-based.... 5-2
VME.... 5-7
broadcast.... 5-8
short address.... 5-7
Executing a Download (PD)....A-23
Executing an Upload (PD)....A-24
Exponential functions.... 5-16
Export User Label File.... 4-15
expressions.... 5-15
evaluation.... 5-15
operators.... 5-16
Precedence.... 5-15
type conversion.... 5-15
Extended Diagnostic.... 3-32
Feedback Capture Status.... 3-108
Feedback Position.... 3-101
Feedback Velocity.... 3-102
Find, Find Next.... 4-10
Finish6-40
First Pass Errors.... 8-21
Follow axis feedback.... 7-33
Format of Data Sent to the CLC....A-4
Function arguments....5-10, 6-79
FUNCTION/ARG.... 7-43
FUNCTION/END.... 7-44
FUNCTION/START.... 7-45
geometry segments.... 5-20
Go.... 6-41
GOSUB (Go To Subroutine).... 7-46
GOTO (Go To Mark).... 7-47
Help.... 4-73
Home.... 6-42
Host Serial Port Setup.... 3-14
I/O.... 6-43
I/O Binary Forcing State (RS)....A-35
I/O Bit Forcing.... 2-6
I/O Device Direction.... 3-16
I/O Device Setup.... 3-16
I/O Device VME Start Address.... 3-16
I/O Forcing Selection (RF)....A-34
I/O Forcing State Change (RC)....A-34
I/O Mapper.... 4-41
I/O Mapper String Format.... 2-4
I/O Register Access (RB), (RX), (RD)...A-33
I/O Register Read....A-33
I/O Register Write....A-33

I/O Setup 4-20
Icon Language Warnings and Error Messages 8-18
icon palette 6-1
ICON PROGRAMMING 6-1
Icons
Accel 6-10
Axis 6-11
AxisEvt 6-16
Branch 6-17
Calc 6-19
Cam 6-22
Cam Build 6-24
Circle 6-31
Connecting Icons 6-3
Decel 6-33
ELS 6-34
ELSAdj 6-37
ELSMode 6-38
Event 6-39
Finish 6-40
Go 6-41
Home 6-42
I/O 6-43
Join 6-44
Joint 6-45
Line 6-46
List of CLC Icons 6-6
Move 6-47
Msg 6-48
Param 6-49
Path 6-51, 6-52, 6-53
PLS 6-56
Position 6-57
Prmbit 6-58
Prmint 6-59
Probe 6-60
ProbeEvt 6-61
Ratio 6-62
Reg 6-64
Registration 6-65
RobotOrg 6-72
RobotTol 6-72
Scissor 6-73
Sequencer 6-74
Size: 6-76
Start 6-77
Stop 6-78
Sub 6-79
Veloc 6-80
VME
Address 6-82
Address Bus Width 6-81
Byte Order 6-82
Data Bus Width 6-82
Format 6-82
Local data 6-82
Transfer count 6-82
VME: 6-81
VMEEvt 6-83
Wait 6-84
Axis in Position 6-84
Coordinated State 6-85
IF (If-Else-Endif Conditional Branch) 7-48
Import User Label File 4-15
Initializing a Download (PW) A-22
Initializing an Upload (PR) A-24
Input/Output Registers A-32
Instruction Format 7-3
Instruction Pointer at Error 3-82
Integral Action Time D-6
Item name B-2, *B-3*
jerk limiting 5-22
Jogging 4-31
Jogging and Display 3-24
Join 6-44
Joint 6-45
KINEMATIC (Use a Kinematic Definition for a Task) 7-53, 7-54
Kinematic Values 3-78
Labels
Bit 4-14
Export 4-15
Import 4-15
Register 4-13
User 4-11
Language Selection 3-14
Last Active Event Number 3-85
Latch 2-1, 2-27
Line 6-46
Linear Position Units 3-92
List CLC Resident Programs (PH) A-25
List of All Parameters 3-59, 3-85, 3-111
List of Event Function Marks (PF) A-26
List of Required Parameters 3-59, 3-66, 3-67, 3-85, 3-111
List Variable Labels (PV) A-27
Local variables 5-10, 6-79
Look Ahead Distance 3-76
Master Axis for Ratio Function 3-98

Maximum Acceleration 3-96
Maximum Deceleration 3-97
Maximum Jog Increment 3-77, 3-97
Maximum Jog Velocity..... 3-77, 3-97
Maximum Model Deviation..... D-5
Maximum Path Acceleration 3-76
Maximum Path Deceleration.....3-76
Maximum Path Speed 3-76
Maximum Velocity.....3-96
Mechanical D-10
MESSAGE/DIAG (Task Diagnostic Message Definition) 7-55
MESSAGE/STATUS (Task Status Message Definition) 7-56
motion
coordinated 1-7
circular interpolation.....1-8
constant speed 1-8
kinematics..... 1-8
linear interpolation.....1-8
ELS 1-8, C-1
phase synchronous mode.....1-8
velocity synchronous mode.....1-8
non-coordinated 1-7
ratioed axes.....1-7
single axis.....1-7
velocity mode 1-7
Move 6-47
MOVE/CIRCLE (Coordinated Move with Circular Interpolation).....7-57
MOVE/JOINT (Coordinated Move Joint Point to Point).....7-58
MOVE/LINE (Coordinated Move with Straight Line Interpolation).....7-59
Msg 6-48
multi-tasking 5-1
Multi-turn Encoder..... D-8
names
symbolic.....4-11
Numeric Data Formats A-4
Operating Mode.....3-31
OPERATOR DEVICES/INTERFACES .. B-1
operators
logical.....5-16
mathematical..... 5-10, 5-16, 6-20
Oscilloscope.....4-34, D-13
Overview.....4-22
P Data A-15
palette.....6-1
Parallel I/O Device.....3-15
Param.....6-49
Parameter Data Subclass A-9
Parameter Lists..... 3-59, 3-111, A-11
Parameter Transfer Commands 3-2
PARAMETER/BIT (Initialize Parameter Bit)7-60
PARAMETER/GET (Load Parameter to a Variable)..... 7-62
PARAMETER/INIT (Initialize a Parameter)7-63
PARAMETER/SET (Set a Parameter) .. 7-64
Parameters
Axis Feedback Capture (Registration) 3-13
Axis Parameter Lists..... 3-13
Axis Parameters - Electronic Line Shaft3-13
Axis Setup 3-12
Axis Status 3-12
Electronic Line Shaft 3-5, 3-7
Jogging and Display 3-4
Parameter Lists 3-7
Program Management 3-4
System Setup 3-3
System Status 3-4
Task Setup 3-10
Task Status 3-10
VME 3-5
Path 6-51, 6-52, 6-53
path planner 5-1, 5-20
acceleration/deceleration 5-22
blend segments 5-21
geometry segments 5-20
jerk limiting..... 5-22
speed..... 5-22
PATH/ABORT (Aborts Coordinated Motion)7-65
PATH/POSITION (Get Current Path Absolute Position)..... 7-66
PATH/RESUME (Resume Coordinated Motion) 7-67
PATH/STOP (Halt Coordinated Motion) 7-68
PATH/WAIT (Pause Program for Motion)7-69
PC Backplane Communications.....*B-8*
Pcam 6-24, C-12
Pendant Security..... 4-23
Pendant Serial Port Setup..... 3-15
Phase Adjusted Status C-27
Phase synchronization1-9, *C-3*, C-4, C-5, C-9
PID Control Loops 4-45
PLC/CLEAR (Clear I/O Register Bit)7-70, 7-71
PLC/READ (Read I/O Register(s)) 7-72
PLC/SET (Set I/O Register Bit) 7-73
PLC/TEST (Test I/O Register Bit) 7-74
PLC/WAIT (Pause Program for I/O)..... 7-75
PLC/WRITE (Write to I/O Register(s) ... 7-76

PLS 4-51, 6-56
PLS/INIT (Compiler directive)..... 7-77
Point Table Data, Row Format A-18
Point Tables A-17
Point Table Data A-18
Point Table Data, Row Format A-18
Points 4-58
Position..... 6-57
Positioning Options 3-88
Positioning Using Secondary Encoder ... 3-91
Print 4-8
PrmBit 6-58
Prmint..... 6-59
Probe 6-60
Probe 1 Negative Captured Position..... 3-109
Probe 2 Positive Captured Position 3-109
ProbeEvt..... 6-61
Program Communication..... A-22
Activating a Program (PA)..... A-25
Download Block Size..... A-22
Erase All Forcing Masks (RE)..... A-35
Erasing (Deleting) a Program (PE) A-24
Executing a Download (PD) A-23
Executing an Upload (PD)..... A-24
I/O Binary Forcing State (RS) A-35
I/O Forcing Selection (RF) A-34
I/O Forcing State Change (RC)..... A-34
I/O Register Access (RB), (RX), (RD) A-33
Initializing a Download (PW) A-22
Initializing an Upload (PR)..... A-24
Input/Output Registers..... A-32
List CLC Resident Programs (PH)..... A-25
List of Event Function Marks (PF)..... A-26
List Variable Labels (PV)..... A-27
Program Header Record A-22
Request Currently Active Program (PA) A-26
Request Name of Program (PN)..... A-26
Set Current I/O State with Mask (RM) A-33
Transfer Tables Between Programs (PX) A-28, A-30
Program Header Record A-22
Program Management..... 3-26, 4-2
program tasks..... 5-1
Programmed Acceleration 3-102
Programmed Velocity 3-101
Proportional Gain..... D-6
Rate Limit Status 3-80
Ratio 6-62
Ratio Slave 3-87
Reading and Writing Physical I/O 2-6
Reading Data from the CLC..... A-2
Real master through SERCOS..... 7-33
Real master to each slave..... 7-33
Real Master Zero Velocity Window..... 3-35
Real Masters..... C-1
Reg 6-64
Register Labels..... 4-13
Registers 4-60
Registration..... 4-63, 6-65
registration error..... 6-65
relative point table 5-23
Relative Synchronization..... C-4
Request Currently Active Program (PA) A-26
Request Name of Program (PN)..... A-26
Required DDS Setup 3-113
RETURN (Return From Subroutine) 7-79, 7-80
ROBOT/ORIGIN 7-80
ROBOT/TOOL 7-81
Robotics..... 3-78
RobotOrg 6-72
RobotTol 6-72
SCAM C-16
Scissor..... 6-73
scissor icon..... 6-3, 6-46
Second Pass Compiler Errors 8-21
Segment Status 3-79
SEQ/LIST 7-83
SEQ/STEP 7-84
Sequencer 4-64, 5-11, 6-74
List 5-11
Single Stepping 5-12
Step List 5-11
Sequencer 7-82
SERCOS 1-4
Data..... 3-109
SERCOS Communication Phase 3-31
SERCOS Control Word..... 3-103
SERCOS Status Word 3-103
Serial Communications B-8
Service name B-2
Set Current I/O State with Mask (RM) ... A-33
Setup Menu 4-17
Show Program Flow..... 4-35
Single Axis..... 3-87
Single-Axis Status Word..... 3-103
Size 6-76
Slave Secondary Mode *C-2*, C-9
Smoothing Time Constant D-6
speed 5-22
Spline..... 6-24

Start 6-77
Stop 6-78
Sub 6-79
Subroutines 4-16, 5-10
symbolic names.....4-11
Synchronization Status.....C-6
System 4-70
System Control Register.....2-8
System Parameters
Parameter Lists..... 3-59, 3-111
System Parameters
Electronic Line Shaft.....3-33
C-0-0150 ELS Master Type.....3-33
C-0-0151 ELS Master Axis.....3-33
C-0-0152 ELS Master Task Assignment 3-33
C-0-0153 Virtual Master Programmed Velocity.....3-34
C-0-0154 Virtual Master Programmed Acceleration.....3-34
C-0-0155 Virtual Master Programmed Deceleration 3-34
C-0-0156 Virtual Master E-Stop Deceleration 3-34
C-0-0157 ELS Master Current Position3-35
C-0-0158 ELS Master Current Velocity3-35
C-0-0159 Real Master Zero Velocity Window 3-35
C-0-0160 Virtual Master Maximum Jog Velocity.....3-35
Jogging and Display.....3-24
C-0-0042 World Large Increment....3-24
C-0-0043 World Small Increment....3-24
C-0-0045 World Fast Jog Speed.....3-24
C-0-0046 World Slow Jog Speed....3-25
C-0-0052 Axis Large Increment.....3-25
C-0-0053 Axis Small Increment.....3-25
C-0-0055 Axis Fast Jog Velocity.....3-25
C-0-0056 Axis Slow Jog Velocity.....3-25
Parameter Lists
C 1.2000 List of All Parameters3-85, 3-111
C 1.2001 List of Required Parameters3-66, 3-67, 3-85, 3-111
C-0-2000 List of All Parameters 3-59
C-0-2001 List of Required Parameters3-59
C-0-3100 Cam Tags.....3-69
Program Management.....3-26
C-0-0091 Total Program Memory 3-26
C-0-0092 Available Program Memory3-26
C-0-0093 Contiguous Program Memory3-26
System Setup
C 1.1 Language Selection.....3-14
C 1.2 Unit Number 3-14
C 1.3 Host Serial Port Setup..... 3-14
C 1.4 Pendant Serial Port Setup 3-15
C 1.5 Parallel I/O Device..... 3-15
C 1.6 I/O Device Setup 3-16
C 1.7 I/O Device Direction..... 3-16
C-0-0008 I/O Device VME Start Address 3-16
C-0-0009 Error Reaction Mode 3-17
System Status 3-29
C-0-0120 Operating Mode 3-31
C-0-0121 SERCOS CommunicationPhase 3-31
C-0-0124 Extended Diagnostic 3-32
C-0-0125 System Timer Value..... 3-32
VME Bus 3-37
C-0-0200 Bus Arbitration Mode..... 3-37
C-0-0201 Bus Release Mode..... 3-37
C-0-0202 VME/CLC Interface..... 3-38
C-0-0203 VME Arbiter Status 3-39
C-0-0232 Unit 4 Window Address Page3-43
C-0-0287 Unit F Window Address Page 3-59
System Status..... 3-29
System Status Register 2-1, 2-18
System Timer Value..... 3-32
T Label Text.....A-15
tables 5-17
event..... 5-19
indirect addressing 5-18
points
absolute..... 5-22
relative..... 5-23
zone protection 5-24
Tagnames.....B-9
Target Point Number 3-79
Target Position..... 3-101
Task Control Registers 2-10
Task Diagnostic Message..... 3-81
Task Jog Control Registers 2-14
Coordinated Jogging:..... 2-14
Task Motion Type 3-72
Task Operating Mode 3-81
Task Parameter Lists..... 3-85
Task Parameters 3-72
Task Setup 3-72
T-0-0001 Task Motion Type..... 3-72
T-0-0005 World Position Units..... 3-74
T-0-0011 Coordinated X Axis..... 3-75
T-0-0012 Coordinated Y Axis..... 3-75
T-0-0013 Coordinated Z Axis..... 3-75

T-0-0020 Maximum Path Speed.....3-76
T-0-0021 Maximum Path Acceleration3-76
T-0-0022 Maximum Path Deceleration3-76
T-0-0023 Look Ahead Distance.....3-76
T-0-0024 Velocity Override.....3-77
T-0-0025 Maximum Jog Increment.3-77
T-0-0026 Maximum Jog Velocity 3-77
T-0-0050 Kinematic Value 1 3-78
T-0-0051 Kinematic Value 2 3-78
T-0-0052 Kinematic Value 3 3-78
T-0-0053 Kinematic Value 4 3-78
T-0-0054 Kinematic Value 5 3-79
T-0-0055 Kinematic Value 6 3-79
T-0-0056 Kinematic Value 7 3-79
T-0-0057 Kinematic Value 8 3-79
T-0-0058 Kinematic Value 9 3-79
Task Status.....3-79
T-0-0100 Target Point Number.....3-79
T-0-0101 Segment Status 3-79
T-0-0102 Rate Limit Status.....3-80
T-0-0111 Current X Position.....3-80
T-0-0112 Current Y Position.....3-81
T-0-0113 Current Z Position.....3-81
T-0-0120 Task Operating Mode 3-81
T-0-0122 Task Diagnostic Message3-81
T-0-0123 Task Status Message 3-82
T-0-0130 Current Instruction Pointer3-82
T-0-0131 Current Instruction 3-82
T-0-0132 Instruction Pointer at Error3-82
T-0-0200 Last Active Event Number3-85
Task Setup.....3-72
Task Status 3-79, 3-81
Task Status Message.....3-82
Task Status Registers.....2-19
TASK/AXES (Task Axes Definition).....7-85
TASK/END (Mark the End of a Task) 7-87
TASK/START & TASK/END 7-3
TASK/START (Define the Start of a Task(s)) 7-88
tasks 1-6, 4-70
Teach Pendant
Control 2-1, 2-29
Status..... 2-1, 2-28
Test 8-2
Text Language Error Messages 8-20
Text Language Programming
CLC Instructions
AXIS/EVENT.....7-6
AXIS/HOME 7-7
AXIS/INITIALIZE 7-8
AXIS/MOVE (Single Axis, Non-Coordinated) 7-10
AXIS/RATIO 7-11
AXIS/SPINDLE (Continuous Velocity Mode) 7-13
AXIS/START..... 7-15
AXIS/STOP..... 7-16
AXIS/WAIT (Axis Wait For In-Position)7-17
Call..... 7-18
CAM/ACTIVATE..... 7-19
CAM/ADJUST 7-20
CAM/BUILD 7-21
CAM/STATUS 7-22, 7-23
CAPTURE/ENABLE..... 7-24
CAPTURE/SETUP 7-25
DATA/SIZE (Configure Memory Allocations) 7-26
DEFINE (Define Name For Variable)7-28
DELAY (Suspend Task Execution) . 7-29
ELS/ADJUST (Adjust ELS Axis) 7-30
ELS/DECEL (Set Master Decel) 7-31
ELS/INIT (Initialize ELS Axes) 7-32
ELS/MODE (Set ELS Axis Mode)... 7-34
ELS/STOP 7-35
EVENT/DONE (Signal Event Completed) 7-36
EVENT/END (Mark End of Event) .. 7-38
EVENT/START (Start of Event function) 7-39
EVENT/TRIGGER (Trigger a Task Event) 7-40
EVENT/WAIT (Pause Task for Event Done Signal)..... 7-42
FUNCTION/ARG..... 7-43
FUNCTION/END 7-44
FUNCTION/START..... 7-45
GOSUB (Go To Subroutine)..... 7-46
GOTO (Go To Mark)..... 7-47
IF (If-Else-Endif Conditional Branch)7-48
Instruction Format 7-3
KINEMATIC (Use a Kinematic Definition for a Task)..... 7-53, 7-54
MESSAGE/DIAG (Task Diagnostic Message Definition)..... 7-55
MESSAGE/STATUS (Task Status Message Definition)..... 7-56
MOVE/CIRCLE (Coordinated Move with Circular Interpolation)..... 7-57
MOVE/JOINT (Coordinated Move Joint Point to Point) 7-58
MOVE/LINE (Coordinated Move with Straight Line Interpolation) 7-59
PARAMETER/BIT (Initialize Parameter Bit) 7-60

PARAMETER/GET (Load Parameter to a Variable) 7-62
PARAMETER/INIT (Initialize a Parameter) 7-63
PARAMETER/SET (Set a Parameter) 7-64
PATH/ABORT (Aborts Coordinated Motion) 7-65
PATH/POSITION (Get Current Path Absolute Position) 7-66
PATH/RESUME (Resume Coordinated Motion) 7-67
PATH/STOP (Halt Coordinated Motion) 7-68
PATH/WAIT (Pause Program for Motion) 7-69
PLC/CLEAR (Clear I/O Register Bit) 7-70, 7-71
PLC/READ (Read I/O Register(s)) .. 7-72
PLC/SET (Set I/O Register Bit) 7-73
PLC/TEST (Test I/O Register Bit) ... 7-74
PLC/WAIT (Pause Program for I/O) 7-75
PLC/WRITE (Write to I/O Register(s) 7-76
PLS/INIT (Compiler directive) 7-77
RETURN (Return From Subroutine) 7-79, 7-80
ROBOT/ORIGIN 7-80
ROBOT/TOOL 7-81
SEQ/LIST 7-83
SEQ/STEP 7-84
SEQUENCER 7-82
TASK/AXES (Task Axes Definition) 7-85
TASK/END (Mark the End of a Task) 7-87
TASK/START (Define the Start of a Task(s)) 7-88
VME/BROADCAST (Broadcast Event to VME) 7-90
VME/EVENT (Send Event to VME Unit) 7-91
VME/READ (VME Bus Data Read) . 7-92
VME/WRITE (VME Bus Data Write) 7-94
Directives 7-1
DEFINE 7-2
EQU (Equate) 7-1
EVENT/START & EVENT/END 7-2
TASK/START & TASK/END 7-3
The Status Menu 4-69
The Tools Menu 4-26
Topic name B-2, *B-8*
Torque Feedback (cyclic) 3-104
Torque Mode Commanded Torque 3-104
Total Program Memory 3-26
Transcendental functions 5-16
Transfer Cams 4-5
Transfer Events 4-6
Transfer I/O Mapper 4-6
Transfer Parameters 4-6
Transfer Points 4-7
Transfer Tables Between Programs (PX) A-28, A-30
Transfer Variables 4-8
Transfer Zones 4-8
Unit 4 Window Address Page 3-43
Unit F Window Address Page 3-59
Unit Number 3-14
Unlatch 2-27
User Labels 4-12
User Program Variables A-15
Variables 4-67, 5-13
global 5-14
VCAM C-13
Veloc 6-80
Velocity Mode 3-87
Velocity Mode Ramp 3-90
Velocity Override 3-77
velocity profiles C-5
Velocity synchronization 1-8, C-4
View Menu 4-16
Virtual Master 6-34
Virtual Master E-Stop Deceleration 3-34
Virtual Master Maximum Jog Velocity 3-35
Virtual Master Programmed Acceleration 3-34
Virtual Master Programmed Deceleration 3-34
Virtual Master Programmed Velocity 3-34
Virtual master through SERCOS 7-33
Virtual Masters C-2
VisualMotion
Options Menu 4-72
Parameters Menu
Analog Outputs D-3
Drive Homing D-8
Drive Reference D-8
Drive Tuning D-6
Mechanical D-10
Setup Menu
Variables 4-67
VisualMotion 1-5
Edit Menu 4-10
File Menu 4-2
Print 4-8
Menu Commands 4-1
VME 6-81
VME Arbiter Status 3-39
VME Backplane Communications *B-8*
VME Bus 3-37
VME Configure 4-24

VME/BROADCAST (Broadcast Event to VME) 7-90
VME/CLC Interface 3-38
VME/EVENT (Send Event to VME Unit) 7-91
VME/READ (VME Bus Data Read)..... 7-92
VME/WRITE (VME Bus Data Write) 7-94
VMEEvt 6-83
Wait 6-84
World Fast Jog Speed..... 3-24
World Large Increment 3-24
World Position Units 3-74
World Slow Jog Speed 3-25
World Small Increment 3-24
Writing Data to the CLC..... A-2
zone protection table 5-24
Zones..... 4-68, 5-24
Zoom Out..... 4-16

Customer Service

Americas (United States, Canada, and Latin America)

USA	USA	INDRAMAT Service Hotlines	Canada
Rexroth Corporation INDRAMAT Division 5150 Prairie Stone Parkway Hoffman Estates, Illinois 60192 Phone: 847/645-36 00 Fax: 857/645-62 01	Rexroth Corporation INDRAMAT Division 2110 Austin Avenue Rochester Hills, Michigan 48309 Phone: 810/853-82 90 Fax: 810/853-82 90	USA: 1-800-860-1055 Canada: 1-847-645-3600 (request Service Department) Latin America: (US Phone No.) 847-645-3600 (request Service Department)	Basic Technologies Corporation Burlington Division 3426 Mainway Drive Burlington, Ontario Canada L7M 1A8 Phone: 905/335-55 11 Fax: 905/335-41 84
Mexico	**Argentina**	**Argentina**	**Brazil**
Motorización y Diseño de Controles, S.A. de C.V. Av. Dr. Gustavo Baz No. 288 Col. Parque Industrial la loma Apartado Postal No. 318 54060 Tlalnepantla Estado de Mexico Mexico Phone: 5/397 86 44 Fax: 5/398 98 88	Mannesmann Rexroth S.A.I.C. Division INDRAMAT Acassusso 48 41/7 1605 Munro (Buenos Aires) Argentina Phone: 01/756 01 40 01/756 02 40 Telex: 262 66 rexro ar Fax: 01/756 01 36	Nakase Asesoramiento Tecnico Diaz Velez 2929 1636 Olivos (Provincia de Buenos Aires) Argentina Phone: 01/790 52 30	Mannesmann Rexroth Automação Ltda. Divisão INDRAMAT Rua Georg Rexroth, 609 Vila Padre Anchieta BR-09.951-250 Diadema-SP Caixa Postal 377 BR-09.901-970 Diadema-SP Phone: 011/745 90 65 011/745 90 70 Fax: 011/745 90 50

Customer Service in the Americas

Asia, Australia, and Far East

Australia	China	China	China
Australian Industrial Machinery Services Pty. Ltd. Unit 3/45 Horne ST Campbellfield VIC 2061 Australia Phone: 03/93 59 0228 Fax: 03/93 59 02886	Rexroth (China) Ltd. Shanghai Office Room 206 Shanghai Intern. Trade Centre 2200 Yanan Xi Lu Shanghai 200335 P.R. China Phone: 021/627 55 333 Fax: 021/627 55 666	Rexroth (China) Ltd. Shanghai Parts & Service Centre 199 Wu Cao Road, Hua Cao Minhang District Shanghai 201 103 P.R. China Phone: 021/622 00 058 Fax: 021/622 00 068	Rexroth (China) Ltd. 1430 China World Trade Centre 1, Jianguomenwai Avenue Beijing 100004 P.R. China Phone: 010/50 50 380 Fax: 010/50 50 379
China	**Hong Kong**	**India**	**Japan**
Rexroth (China) Ltd. A-5F., 123 Lian Shan Street Sha He Kou District Dalian 116 023 P.R. China Phone: 0411/46 78 930 Fax: 0411/46 78 932	Rexroth (China) Ltd. 19 Cheung Shun Street 1st Floor, Cheung Sha Wan, Kowloon, Hong Kong Phone: 741 13 51/-54 or 741 14 30 Telex: 3346 17 GL REX HX Fax: 786 40 19 786 07 33	Mannesmann Rexroth (India) Ltd. INDRAMAT Division Plot. 96, Phase III Peenya Industrial Area Bangalore - 560058 India Phone: 80/839 21 01 80/839 73 74 Telex: 845 5028 RexB Fax: 80/839 43 45	Rexroth Co., Ltd. INDRAMAT Division I.R. Building Nakamachidai 4-26-44 Tsuzuki-ku, Yokohama 226 Japan Phone: 045/942-72 10 Fax: 045/942-03 41
Korea	**Korea**		
Rexroth-Seki Co Ltd. 1500-12 Da-Dae-Dong Saha-Gu, Pusan, 604-050 Korea Phone: 051/264 90 01 Fax: 051/264 90 10	Seo Chang Corporation Ltd. Room 903, Jeail Building 44-35 Yoido-Dong Youngdeungpo-Ku Seoul, Korea Phone: 02/780-82 07 ~9 Fax: 02/784-54 08		

Customer Service in Asia, Australia, and the Far East

Europe

Austria	Austria	Belgium	Denmark
G.L.Rexroth Ges.m.b.H. Geschäftsbereich INDRAMAT Hägelingasse 3 A-1140 Wien Phone: 1/9852540-400 Fax: 1/9852540-93	G.L.Rexroth Ges.m.b.H. Geschäftsbereich INDRAMAT Randlstraße 14 A-4061 Pasching Phone: 07229/4401-36 Fax: 07229/4401-80	Mannesmann Rexroth N.V.-S.A. INDRAMAT Division Industrielaan 8 B-1740 Ternat Phone: 02/5823180 Fax: 02/5824310	BEC Elektronik AS Zinkvej 6 DK-8900 Randers Phone: 086/447866 Fax: 086/447160
England	**Finland**	**France**	**France**
Mannesmann Rexroth Ltd. INDRAMAT Division 4 Esland Place, Love Lane Cirencester, Glos GL7 1YG Phone: 01285/658671 Fax: 01285/654991	Rexroth Mecman OY Riihimiehentie 3 SF-01720 Vantaa Phone: 0/848511 Fax: 0/846387	Rexroth - Sigma S.A. Division INDRAMAT Parc des Barbanniers 4, Place du Village F-92632 Gennevilliers Cedex Phone: 1/41475430 Fax: 1/47946941	Rexroth - Sigma S.A. Division INDRAMAT 91, Bd 1 Joliot Curie F-69634 Venissieux - Cx Phone: 78785256 Fax: 78785231
France	**Italy**	**Italy**	**Netherlands**
Rexroth - Sigma S.A. Division INDRAMAT 270, Avenue de lardenne F-31100 Toulouse Phone: 61499519 Fax: 61310041	Rexroth S.p.A. Divisione INDRAMAT Via G. Di Vittoria, 1 I-20063 Cernusco S/N.MI Phone: 02/92365-270 Fax: 02/92108069	Rexroth S.p.A. Divisione INDRAMAT Via Borgomanero, 11 I-10145 Torino Phone: 011/7712230 Fax: 011/7710190	Hydraudyne Hydrauliek B.V. Kruisbroeksestraat 1a P.O. Box 32 NL-5280 AA Boxtel Phone: 04116/51951 Fax: 04116/51483
Spain	**Spain**	**Sweden**	**Switzerland**
Rexroth S.A. Centro Industrial Santiago Obradors s/n E-08130 Santa Perpetua de Mogoda (Barcelona) Phone: 03/718 68 51 Telex: 591 81 Fax: 03/718 98 62	Goimendi S.A. División Indramat Jolastokieta (Herrera) Apartado 11 37 E-San Sebastion, 20017 Phone: 043/40 01 63 Telex: 361 72 Fax: 043/39 93 95	AB Rexroth Mecman INDRAMAT Division Varuvägen 7 S-125 81 Stockholm Phone: 08/727 92 00 Fax: 08/64 73 277	Rexroth SA Département INDRAMAT Chemin de l`Ecole 6 CH-1036 Sullens Phone: 021/731 43 77 Fax: 021/731 46 78
Switzerland	**Russia**		
Rexroth AG Geeschäftsbereich INDRAMAT Gewerbestraße 3 CH-8500 Frauenfeld Phone: 052/720 21 00 Fax: 052/720 21 11	Tschudnenko E.B. Arsenia 22 153000 Ivanovo Russia Phone: 093/22 39 633		

Customer Service in Europe

Germany

Central Sales Region	Eastern Sales Region	Western Sales Region	Northern Sales Region
INDRAMAT GmbH Bgm.-Dr.-Nebel-Str. 2 D-97816 Lohr am Main Phone: 09352/40-0 Fax: 09352/40-4885	INDRAMAT GmbH Beckerstraße 31 D-09120 Chemnitz Phone: 0371/3555-0 Fax: 0371/3555-230	INDRAMAT GmbH Hansastraße 25 D-40849 Ratingen Phone: 02102/4318-0 Fax: 02102/41315	INDRAMAT GmbH Fährhausstraße 11 D-22085 Hamburg Phone: 040/227126-16 Fax: 040/227126-15
Southern Sales Region	**Southwestern Sales Region**	**INDRAMAT Service Hotline**	
INDRAMAT GmbH Ridlerstraße 75 D-80339 München Phone: 089/540138-30 Fax: 089/540138-10	INDRAMAT GmbH Böblinger Straße 25 D-71229 Leonberg Phone: 07152/972-6 Fax: 07152/972-727	INDRAMAT GmbH Phone: 0172/660 040 6 -or- Phone: 0171/333 882 6	

Customer Service in Germany

Notes:

Indramat